

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problems Mailbox.**

jc675 U.S. PTO
09/481984
01/11/00

```
/*
 * plugext1.0 is a compiled version of plugext for Tcl.
 * -- Clif Flynt <clif@cflynt.com> Jan 3, 1999
 * --
 * -- Modified from get.c by
 * -- Jean-Claude Wippler <jcw@equi4.com>, September 17, 1998.
 */

#include <stdio.h>
#ifdef _WIN32
#include <windows.h>
#else
#include <dlfcn.h>
#endif
#include <tcl.h>

Tcl_HashTable *dlopen_hashtablePtr;
extern int dummyDebugPrint;

void dlopen_InitHashTable () {
    dlopen_hashtablePtr = (Tcl_HashTable *) ckalloc(sizeof(Tcl_HashTable));
    Tcl_InitHashTable(dlopen_hashtablePtr, TCL_STRING_KEYS);
}

static int
dlopen(ClientData x, Tcl_Interp* interp, int objc, Tcl_Obj*CONST* objv)
{
    char hash_handle[50];
    char *tmp;
    void *handle;
    Tcl_HashEntry *hashEntryPtr;
    Tcl_Obj *returnObjPtr;
    int isNew;

    if (objc > 1)
    {
        tmp = Tcl_GetStringFromObj(objv[1], NULL);
        #if defined (_WIN32)
            handle = LoadLibrary(tmp);
        #else
            handle = dlopen(tmp, RTLD_NOW | RTLD_GLOBAL);
        #endif

        if (!handle) {
            fprintf(stderr, "FAIL IN DLOPEN: %s\n", tmp);
            #if defined (_WIN32)
                fprintf(stderr, "%s\n", "Is there a windows error fcn");
            #else
                fprintf(stderr, "%s\n", dlerror());
            #endif
            return(TCL_ERROR);
        }
    }
}
```

```

/*
 * Allocate the space and initialize the return structure.
 */

sprintf(hash_handle, "X%x", handle);

hashEntryPtr = Tcl_CreateHashEntry(dlopen_hashtablePtr, hash_handle,
&isNew);
Tcl_SetHashValue(hashEntryPtr, handle);

returnObjPtr = Tcl_NewStringObj(hash_handle, -1);

Tcl_SetObjResult(interp, returnObjPtr);

return TCL_OK;
}

__getfpucw() {printf("hit getfpucw\n"); fflush(stdout);}

#ifdef (_WIN32)

static int
dlSym(ClientData x, Tcl_Interp* interp, int objc, Tcl_Obj*CONST* objv) {

    Tcl_Obj *returnObjPtr;

    returnObjPtr = Tcl_NewStringObj("dlSym not implemented for Windows", -1);

    Tcl_SetObjResult(interp, returnObjPtr);
    return TCL_OK;
}

#else

static int
dlSym(ClientData x, Tcl_Interp* interp, int objc, Tcl_Obj*CONST* objv)
{
    char *hash_handle;
    char *tmp;
    void *handle;
    Tcl_HashEntry *hashEntryPtr;
    Tcl_Obj *returnObjPtr;
    void *locat;
    char rtnString[20];

    if (objc > 1)
    {
        hash_handle = Tcl_GetStringFromObj(objv[1], NULL);
        hashEntryPtr = Tcl_FindHashEntry(dlopen_hashtablePtr, hash_handle);
        if (hashEntryPtr == (Tcl_HashEntry *) NULL) {
            char errString[80];
            Tcl_Obj *errCodePtr;

            /*

```

```

    * Define an error code from an integer, and set errorCode.
    */
    errCodePtr = Tcl_NewIntObj(554);
    Tcl_SetObjErrorCode(interp, errCodePtr);

    /*
     * This string will be placed in the global variable errorInfo
     */

    sprintf(errString, "Hash object \"%s\" does not exist.", hash_handle);
    Tcl_AddErrorInfo(interp, errString);

    /*
     * This string will be returned as the result of the command.
     */
    Tcl_AppendResult(interp, "can not find hashed object named \"",
        hash_handle, "\"", (char *) NULL);
    return TCL_ERROR;
}

handle = (char *)Tcl_GetHashValue(hashEntryPtr);

tmp = Tcl_GetStringFromObj(objv[2], NULL);

locat = dlsym(handle, tmp);

sprintf(rtnString, "%x", locat);

}

/*
 * Allocate the space and initialize the return structure.
 */

returnObjPtr = Tcl_NewStringObj(rtnString, -1);

Tcl_SetObjResult(interp, returnObjPtr);

return TCL_OK;
}
#endif

#if defined (_WIN32)

    #define WINEXPORT(t) __declspec(dllexport) t
#elif defined (_WIN)
    #define WINEXPORT(t) t __export
#else
    #define WINEXPORT(t) t
#endif

#if defined (__MWERKS__)
    #pragma export on
#endif

```

```
WINEXPORT(int) Dlopen_Init(Tcl_Interp* interp)
{
    Tcl_CreateObjCommand(interp, "dlopen", dlOpen, 0, 0);
    Tcl_CreateObjCommand(interp, "dlsym", dlSym, 0, 0);

    dlopen_InitHashTable ();

    Tcl_PkgProvide(interp, "dlopen", "1.0");

    return TCL_OK;
}

WINEXPORT(int) Dlopen_SafeInit(Tcl_Interp* interp)
{
    return Dlopen_Init(interp);
}
```

```

package require plugext

#
#
# Embed - state variable
# Embed(uniq) - A unique number for the instances
# Embed(UPPER.$id) - A keyword/value pair for an instance.
# Embed(lower.$id) - A value assigned by the embed command.
#
# Embed(MIMETYPE) - The command for this creation of the plugin
# Embed(instance.$id) - The command for this instance of the plugin
# Embed(canvas.$id) - The canvas for this instance of the plugin
#
# Adds one new arg to embed xx=yy -
# KEEPOPEN=1 default - the embed call does not close the stream after
# sending data to the plugin.
# KEEPOPEN=0 default - the embed call closes the stream after
# sending data to the plugin. Closing the stream causes some
# plugins to start.

namespace eval EMBED {
variable Embed
variable output [open argh w]

set Embed(uniq) 1
set Embed(CanvasParent) ""

switch $tcl_platform(platform) {
    "unix" {
        package require dlopen
        set Embed(xref) {
            { .mid audio/x-midi /usr/lib/netscape/plugins/ump.so }
            { .midi audio/x-midi /usr/lib/netscape/plugins/ump.so }
        }
    }
    "windows" {
        set Embed(xref) {
            { .rpm audio/x-pn-realaudio-plugin ./nppl3260.dll }
            { .rm audio/x-pn-realaudio-plugin ./nppl3260.dll }
            { .spl application/futuresplash ./npswf32.dll }
            { .swf application/shockwave-flash ./npswf32.dll }
            { .pdf application/pdf ./nppdf32.dll }
            { .afl video/animaflex ./nprub.dll }
            { .ivf video/x-ivf ./npindeo.dll }
            { .wav audio/x-wav ./npwave.dll }
            { .mid audio/x-midi ./midid.dll }
            { .midi audio/x-midi ./midid.dll }
            { .npw application/x-npwrap ./NPWrap.dll }
            { .mid audio/x-midi ./npmidi.dll }
            { .midi audio/x-midi ./npmidi.dll }
        }
    }
}

}

proc getCanvas {id} {
    variable Embed
    if {[info exists Embed(canvas.$id)]} {

```

```

        return $Embed(canvas.$id)
    } else {
        return ""
    }
}

```

```

proc getInstance {id} {
    variable Embed
    if {[info exists Embed(canvas.$id)]} {
        return $Embed(instance.$id)
    } else {
        return ""
    }
}

```

```

proc setCanvasParent {windowID} {
    set Embed(CanvasParent) $windowID
}

```

```

proc embed {args} {
    global tcl_platform
    variable Embed
    variable output

    set id $Embed(uniq)
    incr Embed(uniq)

    set mode "EMBED"
    set Embed(HIDDEN.$id) true
    set Embed(KEEPOPEN.$id) 1

    #
    # Parse the args.
    # Find the key=val and key="val" pairs and convert them to
    # set Embed(KEY) val
    # commands.
    #

    foreach arg $args {
        if {[string first "=" $arg] < 0} {continue}
        foreach {key val} [split $arg "="] {
            set val [string trim $val {""]}
            set key [string toupper $key]
        }
        eval [list set Embed($key.$id) $val]
    }

    #
    # If the plugext extension is not loaded, load it.
    # If *n[ui]x platform, dlopen the required X libraries.
    #

    if {[string match "" [info command plugext]]} {
        if {[string match $tcl_platform(platform) unix]} {
            puts $output "loading dlopen, etc" ; flush $output
            # load "./dlopen"

```

```

        dlopen "/usr/X11R6/lib/libXt.so"
        dlopen "/usr/local/lib/libXm.so"
    }
    # load plugext
    plugext debug 1
}

#
# Figure out which plugin we need. Start by checking
# the SRC=...ext to see if we've got that extension in our list.
#

if {[info exists Embed(SRC.$id)]} {
    set ext [file extension $Embed(SRC.$id)]
    set pos [lsearch $Embed(xref) *${ext}*]
    if {$pos >= 0} {
        set pluginPath [lindex [lindex $Embed(xref) $pos] 2]
        if {[info exists Embed(TYPE.$id)]} {
            set Embed(TYPE.$id) [lindex [lindex $Embed(xref) $pos] 1]
        }
    }
}

# If no SRC matched, see if we can get the required info
# via the Mime TYPE=xx value

if {[info exists pluginPath] && [info exists Embed(TYPE.$id)]} {
    set pos [lsearch $Embed(xref) *$Embed(TYPE.$id)*]
    if {$pos < 0} {break}
    set pluginPath [lindex [lindex $Embed(xref) $pos] 2]
}

# If we still can't figure out what to use, give up.

if {[info exists pluginPath]} {
    error "No plugin defined for $args"
}

# Load the required plugin.
# Save it's handle in a state variable keyed by the Mime Type

set pluginID $Embed(TYPE.$id)

if {[info exists Embed($pluginID)]} {
    set Embed($pluginID) [plugext create TST2 -path $pluginPath]
}

foreach a $args {
    if {[string first "=" $a] > 0} {
        if {[info exists embedargs]} {
            append embedargs " " $a
        } else {
            set embedargs $a
        }
    }
}
}

```



```

set Embed(instance.$id) [$Embed($pluginID) new \
    -mime $Embed(TYPE.$id) -embed $embedargs \
    -mode $mode]

#
# Generate a window, if HEIGHT and WIDTH are set, and HIDDEN=false/0
# If HIDDEN == True but HEIGHT/WIDTH are missing, set default size
#    200x200

# Can't use ! with a string, so I kludge to get !hidden
if {"$Embed(HIDDEN.$id)"} {} else {
    set ht 200;
    set wd 200;

    if {[info exists Embed(HEIGHT.$id)]} {
        set ht $Embed(HEIGHT.$id)
    }

    if {[info exists Embed(WIDTH.$id)]} {
        set wd $Embed(WIDTH.$id)
    }

    set Embed(canvas.$id) [canvas $Embed(CanvasParent).cPLUGIN$id \
        -height $ht -width $wd -background yellow]

    puts $output "CANVAS: $Embed(canvas.$id) :: $ht x $wd"; flush $output

    # Grid and update, to make sure the base OS has really allocated
    # space for this window before sending it to the plugin. Else,
    # life gets exciting.

    grid $Embed(canvas.$id) -row 0 -column 0
    update; update idle;

    $Embed(instance.$id) setwindow $Embed(canvas.$id)
    grid forget $Embed(canvas.$id)
}

#
# Figure out the type of stream we'll need to interact with
# Then close the stream again.
#

set Embed(out.$id) [$Embed(instance.$id) open -url file:$Embed(SRC.$id) -mime
$Embed(TYPE.$id) w]
fconfigure $Embed(out.$id) -translation binary
set streamType [$Embed(instance.$id) streamtype]

puts $output "STR: $streamType" ; flush $output

#
# Dump data as required by the stream type.
#

switch $streamType {
    "asfile" {

```

```

        $Embed(instance.$id) streamfile $Embed(SRC.$id)
    }
    "asfileonly" {
        $Embed(instance.$id) streamfile $Embed(SRC.$id)
    }
    "normal" {
        set in [open $Embed(SRC.$id) r]
        fconfigure $in -translation binary

        set maxTransfer [$Embed(instance.$id) writeready $Embed(out.$id)]
        if {$maxTransfer > 64000} {
            set maxTransfer 64000
        }

        puts $output "maxTransfer: $maxTransfer"
        fconfigure $Embed(out.$id) -buffer-size $maxTransfer

        # If the file is small go for speed.
        # If it's greater than 64K, reduce the memory footprint,
        # since it won't be fast no matter what we do.

        if {[file size $Embed(SRC.$id)] > 64000} {
            while {[eof $in]} {
                set dat [read $in 4096]
                puts -nonewline $Embed(out.$id) $dat
            }
            flush $Embed(out.$id)
        } else {
            set dat [read $in]
            close $in
            puts -nonewline $Embed(out.$id) $dat
            flush $Embed(out.$id)
        }
        puts "DONE XFER"
    }
    default {
        error "Unrecognized stream value: $streamType"
    }
}
if {!$Embed(KEEPOPEN.$id)} {
    puts "CLOSING $Embed(out.$id)"
    closeFile $id
}
return $id
}

proc resetWindow {id} {
    variable Embed
    if {"$Embed(HIDDEN.$id)"} {} else {
        $Embed(instance.$id) setwindow $Embed(canvas.$id)
    }
}

proc closeFile {id} {
    variable Embed
    if {[info exists Embed(out.$id)]} {
        close $Embed(out.$id)
        unset Embed(out.$id)
    }
}

```

}
}
}

THIS PAGE BLANK (USPTO)

```

#ifdef _WIN32
#include <windows.h>
#endif
#include "extensionLib.h"

void *getHashData (Tcl_Interp *interp,
                  char *hashName,
                  Tcl_HashTable *hashtablePtr) {
    Tcl_HashEntry *hashEntryPtr;
    hashEntryPtr = Tcl_FindHashEntry(hashtablePtr, hashName);

    if (hashEntryPtr == (Tcl_HashEntry *) NULL) {
        char errString[80];
        Tcl_Obj *errCodePtr;

        /*
         * Define an error code from an integer, and set errorCode.
         */
        errCodePtr = Tcl_NewIntObj(554);
        Tcl_SetObjErrorCode(interp, errCodePtr);

        /*
         * This string will be placed in the global variable errorInfo
         */
        sprintf(errString, "Hash object \"%s\" does not exist.", hashName);
        Tcl_AddErrorInfo(interp, errString);

        /*
         * This string will be returned as the result of the command.
         */
        Tcl_AppendResult(interp, "can not find hashed object named \"",
                        hashName, "\"", (char *) NULL);

        return (void *) NULL;
    }

    /*
     * If we got here, then the search was successful and we can extract
     * the data value from the hash entry and return it.
     */

    return (void *)Tcl_GetHashValue(hashEntryPtr);
}

CmdReturn *makeCmdReturn () {
    CmdReturn *returnStructPtr;

    /*
     * Allocate the space and initialize the return structure.
     */

    returnStructPtr = (CmdReturn *) Tcl_Alloc(sizeof (CmdReturn));
    returnStructPtr->status = TCL_OK;
}

```

```

    returnStructPtr->object = NULL;

    return returnStructPtr;
}

void setErrorReturns(Tcl_Interp *interp,
                    char *formatStr1,
                    char *param1_1,
                    char *param1_2,
                    char *param1_3,
                    char *formatStr2,
                    char *param2_1,
                    char *param2_2,
                    char *param2_3,
                    char *procedureName,
                    char *errString)
{
    char errString1[180];
    char errString2[180];
    char errString3[80];

    sprintf(errString1, formatStr1, param1_1, param1_2, param1_3);
    sprintf(errString2, formatStr2, param2_1, param2_2, param2_3);
    sprintf(errString3, "error in %s", procedureName);

    /*
     * Both of these strings will be added to the Tcl script
     * global variable errorInfo
     */

    Tcl_AddObjErrorInfo(interp, errString1, strlen(errString1));
    Tcl_AddErrorInfo(interp, errString3);

    /*
     * This SetErrorCode command will set the Tcl script
     * variable errorCode to "500"
     */

    Tcl_SetErrorCode(interp, errString, (char *) NULL);

    /*
     * This defines the return string for this subcommand
     */

    Tcl_AppendResult(interp, errString1, (char *) NULL);
}

/*-----
 * void plugext_InitHashTable ()--
 *   plugextinitialize a hash table.
 *   If your application does not need a hash table, this may be deleted.
 *
 * Arguments
 *   NONE
 *
 */

```

```

* Results
*   Initializes the hash table to accept STRING keys
*
* Side Effects:
*   None
-----*/

Tcl_HashTable *initHashTable () {
    Tcl_HashTable *hashtablePtr;

    hashtablePtr = (Tcl_HashTable *) ckalloc(sizeof(Tcl_HashTable));
    Tcl_InitHashTable(hashtablePtr, TCL_STRING_KEYS);
    return hashtablePtr;
}

```

```

#ifndef EXTENSION_LIB_H
#define EXTENSION_LIB_H
#include "tcl805.h"

/*
 * The CmdReturn structure is used by the subroutines to
 * pass back a success/failure code and a Tcl_Obj result.
 *
 * This is not an official Tcl standard return type. I
 * find this works well with commands that accept subcommands.
 */

typedef struct cmd_return {
    int status;
    Tcl_Obj *object;
} CmdReturn;

typedef struct cmd_Def {
    char *usage;
    int minArgCnt;
    int maxArgCnt;
} cmdDefinition;

/* Prototypes for the functions in the extensionLib */

void *getHashData (Tcl_Interp *, char*, Tcl_HashTable*);
CmdReturn *makeCmdReturn();
void setErrorReturns(Tcl_Interp *,
                    char *,      char *,      char *,      char *,
                    char *,      char *,      char *,      char *,
                    char *,      char *);
Tcl_HashTable *initHashTable ();

#endif EXTENSION_LIB_H

```

```

/* -*- Mode: C; tab-width: 4; -*- */
/*****
 * Java Runtime Interface
 * Copyright (c) 1996 Netscape Communications Corporation. All rights reserved.
 *****/

#ifndef JRI_H
#define JRI_H

#include "jri_md.h"
#include <stddef.h>
#include <stdlib.h>
#include <stdarg.h>

#ifdef __cplusplus
extern "C" {
#endif

/*****
 * Types
 *****/

typedef void*                                JRIRef;

typedef struct JRIGlobal*                    JRIGlobalRef;

typedef struct JRINativeInterface    JRINativeInterface;

typedef const JRINativeInterface*    JRIEnv;

typedef struct JRIFieldThunk          JRIFieldThunk;

typedef struct JRIMethodThunk         JRIMethodThunk;

/* convenience types: */
typedef JRIRef      jref;
typedef JRIRef      jbooleanArray;
typedef JRIRef      jbyteArray;
typedef JRIRef      jcharArray;
typedef JRIRef      jshortArray;
typedef JRIRef      jintArray;
typedef JRIRef      jlongArray;
typedef JRIRef      jfloatArray;
typedef JRIRef      jdoubleArray;
typedef JRIRef      jobjectArray;
typedef JRIRef      jstringArray;
typedef JRIRef      jarrayArray;

typedef union JRIValue {
    jbool      z;
    jbyte      b;
    jchar      c;
    jshort     s;
    jint       i;
    jlong      l;
    jfloat     f;
    jdouble    d;
}

```



```

        jref                r;
    } JRIValue;

typedef enum JRIMethodThunkType {
    JRIMethodThunkType_NameAndSig
} JRIMethodThunkType;

struct JRIMethodThunk {
    JRIMethodThunkType    type;
    void*                 data0;
    void*                 data1;
};

typedef enum JRIBoolean {
    JRIFalse              = 0,
    JRITrue                = 1
} JRIBoolean;

typedef struct JRINativeInfo {
    char*                 nativeMethodName; /* input */
    char*                 nativeMethodSig;  /* input */
    void*                 nativeMethodProc; /* input */
} JRINativeInfo;

#define JRIconstructorMethodName    "<init>"

/*****
 * Signature Construction Macros
 *****/

/*
** These macros can be used to construct signature strings. Hopefully their
names
** are a little easier to remember than the single character they correspond to.
** For example, to specify the signature of the method:
**
**     public int read(byte b[], int off, int len);
**
** you could write something like this in C:
**
**     char* readSig = JRISigMethod(JRISigArray(JRISigByte)
**                                     JRISigInt
**                                     JRISigInt) JRISigInt;
**
** Of course, don't put commas between the types.
*/
#define JRISigArray(T)          "[" T
#define JRISigByte              "B"
#define JRISigChar              "C"
#define JRISigClass(name)      "L" name "; "
#define JRISigFloat             "F"
#define JRISigDouble            "D"
#define JRISigMethod(args)     "(" args ")"
#define JRISigNoArgs            ""
#define JRISigInt               "I"
#define JRISigLong              "J"
#define JRISigShort             "S"

```

```

#define JRISigVoid          "V"
#define JRISigBoolean      "Z"

/*****
 * Environments
 *****/

extern JRI_PUBLIC_API(JRIEnv*)
JRI_GetCurrentEnv(void);

/*
 ** LoadClass borrows the buffer that you hand to it, so it will copy it
 ** if it needs to. This is useful when defining a class from static class
 ** data.
 */
typedef jref
(*JRI_LoadClass_t)(JRIEnv* env, const char* buf, jsize bufLen);

/*
 ** Returns a jref to a class object from a fully qualified name (package
 ** names delimited by '/' and the class name). If the name begins with
 ** '[' (the array signature character), an array class is returned.
 */
typedef jref
(*JRI_FindClass_t)(JRIEnv* env, const char* name);

/* Working with Exceptions */

typedef void
(*JRI_Throw_t)(JRIEnv* env, jref obj);

typedef void
(*JRI_ThrowNew_t)(JRIEnv* env, jref clazz, const char* message);

typedef jref
(*JRI_ExceptionOccurred_t)(JRIEnv* env);

typedef void
(*JRI_ExceptionDescribe_t)(JRIEnv* env);

typedef void
(*JRI_ExceptionClear_t)(JRIEnv* env);

/*****
 * References
 *****/

typedef JRIGlobalRef
(*JRI_NewGlobalRef_t)(JRIEnv* env, jref ref);

typedef void
(*JRI_DisposeGlobalRef_t)(JRIEnv* env, JRIGlobalRef ref);

typedef jref
(*JRI_GetGlobalRef_t)(JRIEnv* env, JRIGlobalRef globalRef);

typedef void

```

```

(*JRI_SetGlobalRef_t)(JRIEnv* env, JRIGlobalRef globalRef, jref value);

typedef jbool
(*JRI_IsSameObject_t)(JRIEnv* env, jref r1, jref r2);

/*****
 * Object Operations
 *****/

typedef jref
(*JRI_NewObject_t)(JRIEnv* env, jref clazz, JRIMethodThunk* method, ...);

typedef jref
(*JRI_NewObjectA_t)(JRIEnv* env, jref clazz, JRIMethodThunk* method,
                    JRIValue* args);

typedef jref
(*JRI_NewObjectV_t)(JRIEnv* env, jref clazz, JRIMethodThunk* method,
                    va_list args);

typedef jref
(*JRI_GetObjectClass_t)(JRIEnv* env, jref obj);

typedef jbool
(*JRI_IsInstanceOf_t)(JRIEnv* env, jref obj, jref clazz);

/*****
/* Accessing Public Fields of Objects */

typedef jref
(*JRI_GetField_t)(JRIEnv* env, jref obj, JRIFieldThunk* field);

typedef jbool
(*JRI_GetField_boolean_t)(JRIEnv* env, jref obj, JRIFieldThunk* field);

typedef jbyte
(*JRI_GetField_byte_t)(JRIEnv* env, jref obj, JRIFieldThunk* field);

typedef jchar
(*JRI_GetField_char_t)(JRIEnv* env, jref obj, JRIFieldThunk* field);

typedef jshort
(*JRI_GetField_short_t)(JRIEnv* env, jref obj, JRIFieldThunk* field);

typedef jint
(*JRI_GetField_int_t)(JRIEnv* env, jref obj, JRIFieldThunk* field);

typedef jlong
(*JRI_GetField_long_t)(JRIEnv* env, jref obj, JRIFieldThunk* field);

typedef jfloat
(*JRI_GetField_float_t)(JRIEnv* env, jref obj, JRIFieldThunk* field);

typedef jdouble
(*JRI_GetField_double_t)(JRIEnv* env, jref obj, JRIFieldThunk* field);

```

```

/*****/

typedef void
(*JRI_SetField_t)(JRIEnv* env, jref obj, JRIFieldThunk* field, jref value);

typedef void
(*JRI_SetField_boolean_t)(JRIEnv* env, jref obj, JRIFieldThunk* field, jbool
value);

typedef void
(*JRI_SetField_byte_t)(JRIEnv* env, jref obj, JRIFieldThunk* field, jbyte
value);

typedef void
(*JRI_SetField_char_t)(JRIEnv* env, jref obj, JRIFieldThunk* field, jchar
value);

typedef void
(*JRI_SetField_short_t)(JRIEnv* env, jref obj, JRIFieldThunk* field, jshort
value);

typedef void
(*JRI_SetField_int_t)(JRIEnv* env, jref obj, JRIFieldThunk* field, jint value);

typedef void
(*JRI_SetField_long_t)(JRIEnv* env, jref obj, JRIFieldThunk* field, jlong
value);

typedef void
(*JRI_SetField_float_t)(JRIEnv* env, jref obj, JRIFieldThunk* field, jfloat
value);

typedef void
(*JRI_SetField_double_t)(JRIEnv* env, jref obj, JRIFieldThunk* field, jdouble
value);

/*****/

/* Calling Public Dynamic Methods of Objects */

typedef JRIValue
(*JRI_CallMethod_t)(JRIEnv* env, jref obj, JRIMethodThunk* method, ...);

typedef JRIValue
(*JRI_ApplyMethodV_t)(JRIEnv* env, jref obj, JRIMethodThunk* method, va_list
args);

typedef JRIValue
(*JRI_ApplyMethod_t)(JRIEnv* env, jref obj, JRIMethodThunk* method,
                    JRIValue* valueArray);

/*****
 * Class Operations
 *****/

/*
** Determines whether the first class is a subclass of the second, or

```

```

** whether it has the second class as one of its interfaces.
*/
typedef jbool
(*JRI_IsSubclassOf_t)(JRIEnv* env, jref clazz, jref super);

/*****

/* Accessing Public Static Fields of Objects */

typedef jref
(*JRI_GetStaticField_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field);

typedef jbool
(*JRI_GetStaticField_boolean_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field);

typedef jbyte
(*JRI_GetStaticField_byte_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field);

typedef jchar
(*JRI_GetStaticField_char_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field);

typedef jshort
(*JRI_GetStaticField_short_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field);

typedef jint
(*JRI_GetStaticField_int_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field);

typedef jlong
(*JRI_GetStaticField_long_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field);

typedef jfloat
(*JRI_GetStaticField_float_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field);

typedef jdouble
(*JRI_GetStaticField_double_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field);

*****/

typedef void
(*JRI_SetStaticField_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field,
                        jref value);

typedef void
(*JRI_SetStaticField_boolean_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field,
                                jbool value);

typedef void
(*JRI_SetStaticField_byte_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field,
                              jbyte value);

typedef void
(*JRI_SetStaticField_char_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field,
                              jchar value);

typedef void
(*JRI_SetStaticField_short_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field,
                              jshort value);

```

```

typedef void
(*JRI_SetStaticField_int_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field,
                             jint value);

typedef void
(*JRI_SetStaticField_long_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field,
                              jlong value);

typedef void
(*JRI_SetStaticField_float_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field,
                               jfloat value);

typedef void
(*JRI_SetStaticField_double_t)(JRIEnv* env, jref clazz, JRIFieldThunk* field,
                                jdouble value);

/*****

/* Calling Public Static Methods of Objects */

typedef JRIValue
(*JRI_CallStaticMethod_t)(JRIEnv* env, jref clazz, JRIMethodThunk* method, ...);

typedef JRIValue
(*JRI_ApplyStaticMethodV_t)(JRIEnv* env, jref clazz, JRIMethodThunk* method,
                             va_list args);

typedef JRIValue
(*JRI_ApplyStaticMethod_t)(JRIEnv* env, jref clazz, JRIMethodThunk* method,
                            JRIValue* valueArray);

*****/

/* String Operations
*****/

/* Unicode Interface */

typedef jref
(*JRI_NewString_t)(JRIEnv* env, const jchar* bytes, jsize length);

typedef jsize
(*JRI_GetStringLength_t)(JRIEnv* env, jref string);

/* This returns a const jchar* pointer directly into the string's contents: */
typedef const jchar*
(*JRI_GetStringChars_t)(JRIEnv* env, jref string);

/* Mallocs a C string that must be freed by caller: */
typedef jchar*
(*JRI_CopyString_t)(JRIEnv* env, jref string);

/* UTF Interface */

typedef jref
(*JRI_NewStringUTF_t)(JRIEnv* env, const char* bytes, jsize length);

```

```

typedef jsize
(*JRI_GetStringUTFLength_t)(JRIEnv* env, jref string);

/* Mallocs a UTF C string that must be freed by caller: */
typedef char*
(*JRI_CopyStringUTF_t)(JRIEnv* env, jref string);

/*****
 * Scalar Array Operations
 *****/

typedef jref
(*JRI_NewScalarArray_t)(JRIEnv* env, jsize length, const jbyte* bytes);

typedef jsize
(*JRI_GetScalarArrayLength_t)(JRIEnv* env, jref array);

typedef jbyte*
(*JRI_GetScalarArrayData_t)(JRIEnv* env, jref array);

/*****
 * Object Array Operations
 *****/

typedef jref
(*JRI_NewObjectArray_t)(JRIEnv* env, jsize length, jref elementClass,
                        jref initialElement);

typedef jsize
(*JRI_GetObjectArrayLength_t)(JRIEnv* env, jref array);

typedef jref
(*JRI_GetObjectArrayElement_t)(JRIEnv* env, jref array, jsize index);

typedef void
(*JRI_SetObjectArrayElement_t)(JRIEnv* env, jref array, jsize index,
                               jref value);

/*****
 * Native Bootstrap
 *****/

/*
 ** This routine is used by the generated stub code to look up field
 ** positions and method identifiers for native access and invocation.
 */
typedef void
(*JRI_RegisterNatives_t)(JRIEnv* env, jref clazz,
                        JRINativeInfo* nativeInfoArray);

typedef void
(*JRI_UnregisterNatives_t)(JRIEnv* env, jref clazz);

/*****/

struct JRIFieldThunk {
    void*      data;

```

```

    JRI_GetField_t    get; /* cast this to the right accessor type before
calling */
    JRI_SetField_t    set; /* cast this to the right accessor type before
calling */
};

```

```

struct JRINativeInterface {
    void*                               Reserved0;
    void*                               Reserved1;
    void*                               Reserved2;
    JRI_LoadClass_t                    LoadClass;
    JRI_FindClass_t                    FindClass;
    JRI_Throw_t                        Throw;
    JRI_ThrowNew_t                     ThrowNew;
    JRI_ExceptionOccurred_t             ExceptionOccurred;
    JRI_ExceptionDescribe_t             ExceptionDescribe;
    JRI_ExceptionClear_t               ExceptionClear;
    JRI_NewGlobalRef_t                 NewGlobalRef;
    JRI_DisposeGlobalRef_t             DisposeGlobalRef;
    JRI_GetGlobalRef_t                 GetGlobalRef;
    JRI_SetGlobalRef_t                 SetGlobalRef;
    JRI_IsSameObject_t                 IsSameObject;
    JRI_NewObject_t                     NewObject;
    JRI_NewObjectA_t                   NewObjectA;
    JRI_NewObjectV_t                   NewObjectV;
    JRI_GetObjectClass_t               GetObjectClass;
    JRI_IsInstanceOf_t                 IsInstanceOf;
    JRI_GetField_t                     GetField;
    JRI_GetField_boolean_t             GetField_boolean;
    JRI_GetField_byte_t                GetField_byte;
    JRI_GetField_char_t                GetField_char;
    JRI_GetField_short_t               GetField_short;
    JRI_GetField_int_t                 GetField_int;
    JRI_GetField_long_t                GetField_long;
    JRI_GetField_float_t               GetField_float;
    JRI_GetField_double_t              GetField_double;
    JRI_SetField_t                     SetField;
    JRI_SetField_boolean_t             SetField_boolean;
    JRI_SetField_byte_t                SetField_byte;
    JRI_SetField_char_t                SetField_char;
    JRI_SetField_short_t               SetField_short;
    JRI_SetField_int_t                 SetField_int;
    JRI_SetField_long_t                SetField_long;
    JRI_SetField_float_t               SetField_float;
    JRI_SetField_double_t              SetField_double;
    JRI_CallMethod_t                   CallMethod;
    JRI_ApplyMethodV_t                 ApplyMethodV;
    JRI_ApplyMethod_t                  ApplyMethod;
    JRI_IsSubclassOf_t                 IsSubclassOf;
    JRI_GetStaticField_t               GetStaticField;
    JRI_GetStaticField_boolean_t       GetStaticField_boolean;
    JRI_GetStaticField_byte_t          GetStaticField_byte;
    JRI_GetStaticField_char_t          GetStaticField_char;
    JRI_GetStaticField_short_t         GetStaticField_short;
    JRI_GetStaticField_int_t           GetStaticField_int;
    JRI_GetStaticField_long_t          GetStaticField_long;
    JRI_GetStaticField_float_t         GetStaticField_float;

```



```

JRI_GetStaticField_double_t      GetStaticField_double;
JRI_SetStaticField_t             SetStaticField;
JRI_SetStaticField_boolean_t     SetStaticField_boolean;
JRI_SetStaticField_byte_t        SetStaticField_byte;
JRI_SetStaticField_char_t        SetStaticField_char;
JRI_SetStaticField_short_t       SetStaticField_short;
JRI_SetStaticField_int_t         SetStaticField_int;
JRI_SetStaticField_long_t        SetStaticField_long;
JRI_SetStaticField_float_t       SetStaticField_float;
JRI_SetStaticField_double_t      SetStaticField_double;
JRI_CallStaticMethod_t           CallStaticMethod;
JRI_ApplyStaticMethodV_t         ApplyStaticMethodV;
JRI_ApplyStaticMethod_t          ApplyStaticMethod;
JRI_NewString_t                  NewString;
JRI_GetStringLength_t            GetStringLength;
JRI_GetStringChars_t            GetStringChars;
JRI_CopyString_t                 CopyString;
JRI_NewStringUTF_t               NewStringUTF;
JRI_GetStringUTFLength_t         GetStringUTFLength;
JRI_CopyStringUTF_t              CopyStringUTF;
JRI_NewScalarArray_t             NewScalarArray;
JRI_GetScalarArrayLength_t       GetScalarArrayLength;
JRI_GetScalarArrayData_t         GetScalarArrayData;
JRI_NewObjectArray_t             NewObjectArray;
JRI_GetObjectArrayLength_t       GetObjectArrayLength;
JRI_GetObjectArrayElement_t      GetObjectArrayElement;
JRI_SetObjectArrayElement_t      SetObjectArrayElement;
JRI_RegisterNatives_t            RegisterNatives;
JRI_UnregisterNatives_t          UnregisterNatives;
};

/*****

/*
** LoadClass borrows the buffer that you hand to it, so it will Copy it
** if it needs to. This is useful when defining a class from static class
** data.
*/
#define JRI_LoadClass(env, buf, bufLen) \
    (((*(env))->LoadClass)(env, buf, bufLen))

/*
** Returns a jref to a class object from a fully qualified name (package
** names delimited by '/' and the class name). If the name begins with
** '[' (the array signature character), an array class is returned.
*/
#define JRI_FindClass(env, name) \
    (((*(env))->FindClass)(env, name))

/* Working with Exceptions */

#define JRI_Throw(env, obj) \
    (((*(env))->Throw)(env, obj))

#define JRI_ThrowNew(env, clazz, message) \
    (((*(env))->ThrowNew)(env, clazz, message))

```

```

#define JRI_ExceptionOccurred(env) \
    (((*(env))->ExceptionOccurred) (env))

#define JRI_ExceptionDescribe(env) \
    (((*(env))->ExceptionDescribe) (env))

#define JRI_ExceptionClear(env) \
    (((*(env))->ExceptionClear) (env))

/*****
 * References
 *****/

#define JRI_NewGlobalRef(env, ref) \
    (((*(env))->NewGlobalRef) (env, ref))

#define JRI_DisposeGlobalRef(env, ref) \
    (((*(env))->DisposeGlobalRef) (env, ref))

#define JRI_GetGlobalRef(env, globalRef) \
    (((*(env))->GetGlobalRef) (env, globalRef))

#define JRI_SetGlobalRef(env, globalRef, value) \
    (((*(env))->SetGlobalRef) (env, globalRef, value))

#define JRI_IsSameObject(env, r1, r2) \
    (((*(env))->IsSameObject) (env, r1, r2))

/*****
 * Object Operations
 *****/

#define JRI_NewObjectM(env)      (((*(env))->NewObject)
#define JRI_NewObject          JRI_NewObjectM(env)      /* hard coded 'env'
*/

#define JRI_NewObjectA(env, clazz, method, args) \
    (((*(env))->NewObjectA) (env, clazz, method, args))

#define JRI_NewObjectV(env, clazz, method, args) \
    (((*(env))->NewObjectV) (env, clazz, method, args))

#define JRI_GetObjectClass(env, obj) \
    (((*(env))->GetObjectClass) (env, obj))

#define JRI_IsInstanceOf(env, obj, clazz) \
    (((*(env))->IsInstanceOf) (env, obj, clazz))

/*****
/* Accessing Public Fields of Objects */

#define JRI_GetField(env, obj, fieldThunk) \
    (((JRI_GetField_t) (fieldThunk)->get) (env, obj, fieldThunk))

#define JRI_GetField_boolean(env, obj, fieldThunk) \
    (((JRI_GetField_boolean_t) (fieldThunk)->get) (env, obj, fieldThunk))

```

```

#define JRI_GetField_byte(env, obj, fieldThunk) \
    (((JRI_GetField_byte_t)(fieldThunk)->get)(env, obj, fieldThunk))

#define JRI_GetField_char(env, obj, fieldThunk) \
    (((JRI_GetField_char_t)(fieldThunk)->get)(env, obj, fieldThunk))

#define JRI_GetField_short(env, obj, fieldThunk) \
    (((JRI_GetField_short_t)(fieldThunk)->get)(env, obj, fieldThunk))

#define JRI_GetField_int(env, obj, fieldThunk) \
    (((JRI_GetField_int_t)(fieldThunk)->get)(env, obj, fieldThunk))

#define JRI_GetField_long(env, obj, fieldThunk) \
    (((JRI_GetField_long_t)(fieldThunk)->get)(env, obj, fieldThunk))

#define JRI_GetField_float(env, obj, fieldThunk) \
    (((JRI_GetField_float_t)(fieldThunk)->get)(env, obj, fieldThunk))

#define JRI_GetField_double(env, obj, fieldThunk) \
    (((JRI_GetField_double_t)(fieldThunk)->get)(env, obj, fieldThunk))

/*****/

#define JRI_SetField(env, obj, fieldThunk, value) \
    (((JRI_SetField_t)(fieldThunk)->set)(env, obj, fieldThunk, value))

#define JRI_SetField_boolean(env, obj, fieldThunk, value) \
    (((JRI_SetField_boolean_t)(fieldThunk)->set)(env, obj, fieldThunk, value))

#define JRI_SetField_byte(env, obj, fieldThunk, value) \
    (((JRI_SetField_byte_t)(fieldThunk)->set)(env, obj, fieldThunk, value))

#define JRI_SetField_char(env, obj, fieldThunk, value) \
    (((JRI_SetField_char_t)(fieldThunk)->set)(env, obj, fieldThunk, value))

#define JRI_SetField_short(env, obj, fieldThunk, value) \
    (((JRI_SetField_short_t)(fieldThunk)->set)(env, obj, fieldThunk, value))

#define JRI_SetField_int(env, obj, fieldThunk, value) \
    (((JRI_SetField_int_t)(fieldThunk)->set)(env, obj, fieldThunk, value))

#define JRI_SetField_long(env, obj, fieldThunk, value) \
    (((JRI_SetField_long_t)(fieldThunk)->set)(env, obj, fieldThunk, value))

#define JRI_SetField_float(env, obj, fieldThunk, value) \
    (((JRI_SetField_float_t)(fieldThunk)->set)(env, obj, fieldThunk, value))

#define JRI_SetField_double(env, obj, fieldThunk, value) \
    (((JRI_SetField_double_t)(fieldThunk)->set)(env, obj, fieldThunk, value))

/*****/

/* Calling Public Dynamic Methods of Objects */

#define JRI_CallMethodM(env)          ((*env)->CallMethod)

```

```

#define JRI_CallMethod                                JRI_CallMethodM(env)    /* hard
coded 'env' */

#define JRI_ApplyMethodV(env, obj, methodThunk, args) \
    (((*(env))->ApplyMethodV)(env, obj, methodThunk, args))

#define JRI_ApplyMethod(env, obj, methodThunk, args) \
    (((*(env))->ApplyMethod)(env, obj, methodThunk, args))

/*****
 * Class Operations
 *****/

/*
** Determines whether the first class is a subclass of the second, or
** whether it has the second class as one of its interfaces.
*/
#define JRI_IsSubclassOf(env, clazz, super) \
    (((*(env))->IsSubclassOf)(env, clazz, super))

/*****

/* Accessing Public Static Fields of Objects */

#define JRI_GetStaticField(env, clazz, fieldThunk) \
    (((JRI_GetStaticField_t)(fieldThunk))->get)(env, clazz, fieldThunk))

#define JRI_GetStaticField_boolean(env, clazz, fieldThunk) \
    (((JRI_GetStaticField_boolean_t)(fieldThunk))->get)(env, clazz,
fieldThunk))

#define JRI_GetStaticField_byte(env, clazz, fieldThunk) \
    (((JRI_GetStaticField_byte_t)(fieldThunk))->get)(env, clazz, fieldThunk))

#define JRI_GetStaticField_char(env, clazz, fieldThunk) \
    (((JRI_GetStaticField_char_t)(fieldThunk))->get)(env, clazz, fieldThunk))

#define JRI_GetStaticField_short(env, clazz, fieldThunk) \
    (((JRI_GetStaticField_short_t)(fieldThunk))->get)(env, clazz, fieldThunk))

#define JRI_GetStaticField_int(env, clazz, fieldThunk) \
    (((JRI_GetStaticField_int_t)(fieldThunk))->get)(env, clazz, fieldThunk))

#define JRI_GetStaticField_long(env, clazz, fieldThunk) \
    (((JRI_GetStaticField_long_t)(fieldThunk))->get)(env, clazz, fieldThunk))

#define JRI_GetStaticField_float(env, clazz, fieldThunk) \
    (((JRI_GetStaticField_float_t)(fieldThunk))->get)(env, clazz, fieldThunk))

#define JRI_GetStaticField_double(env, clazz, fieldThunk) \
    (((JRI_GetStaticField_double_t)(fieldThunk))->get)(env, clazz, fieldThunk))

/*****

#define JRI_SetStaticField(env, clazz, fieldThunk, value) \
    (((JRI_SetStaticField_t)(fieldThunk))->set)(env, clazz, fieldThunk, value))

```

```

#define JRI_SetStaticField_boolean(env, clazz, fieldThunk, value) \
    (((JRI_SetStaticField_boolean_t) (fieldThunk)->set) (env, clazz, fieldThunk, value))

#define JRI_SetStaticField_byte(env, clazz, fieldThunk, value) \
    (((JRI_SetStaticField_byte_t) (fieldThunk)->set) (env, clazz, fieldThunk, value))

#define JRI_SetStaticField_char(env, clazz, fieldThunk, value) \
    (((JRI_SetStaticField_char_t) (fieldThunk)->set) (env, clazz, fieldThunk, value))

#define JRI_SetStaticField_short(env, clazz, fieldThunk, value) \
    (((JRI_SetStaticField_short_t) (fieldThunk)->set) (env, clazz, fieldThunk, value))

#define JRI_SetStaticField_int(env, clazz, fieldThunk, value) \
    (((JRI_SetStaticField_int_t) (fieldThunk)->set) (env, clazz, fieldThunk, value))

#define JRI_SetStaticField_long(env, clazz, fieldThunk, value) \
    (((JRI_SetStaticField_long_t) (fieldThunk)->set) (env, clazz, fieldThunk, value))

#define JRI_SetStaticField_float(env, clazz, fieldThunk, value) \
    (((JRI_SetStaticField_float_t) (fieldThunk)->set) (env, clazz, fieldThunk, value))

#define JRI_SetStaticField_double(env, clazz, fieldThunk, value) \
    (((JRI_SetStaticField_double_t) (fieldThunk)->set) (env, clazz, fieldThunk, value))

/*****

/* Calling Public Static Methods of Objects */

#define JRI_CallStaticMethodM(env)          ((*env)->CallStaticMethod)
#define JRI_CallStaticMethod                JRI_CallStaticMethodM(env) /*
hard coded 'env' */

#define JRI_ApplyStaticMethodV(env, clazz, method, args) \
    (((*env)->ApplyStaticMethodV) (env, clazz, method, args))

#define JRI_ApplyStaticMethod(env, clazz, method, args) \
    (((*env)->ApplyStaticMethod) (env, clazz, method, args))

/*****
 * String Operations
 *****/

/* Unicode Interface */

#define JRI_NewString(env, bytes, length) \
    (((*env)->NewString) (env, bytes, length))

#define JRI_GetStringLength(env, string) \
    (((*env)->GetStringLength) (env, string))

```

```

/* This returns a const jchar* pointer directly into the string's contents: */
#define JRI_GetStringChars(env, string) \
    (((*(env))->GetStringChars)(env, string))

/* Mallocs a C string that must be freed by caller: */
#define JRI_CopyString(env, string) \
    (((*(env))->CopyString)(env, string))

/* UTF Interface */

#define JRI_NewStringUTF(env, bytes, length) \
    (((*(env))->NewStringUTF)(env, bytes, length))

#define JRI_GetStringUTFLength(env, string) \
    (((*(env))->GetStringUTFLength)(env, string))

/* Mallocs a UTF C string that must be freed by caller: */
#define JRI_CopyStringUTF(env, string) \
    (((*(env))->CopyStringUTF)(env, string))

/*****
 * Scalar Array Operations
 *****/

#define JRI_NewScalarArray(env, length, bytes) \
    (((*(env))->NewScalarArray)(env, length, bytes))

#define JRI_GetScalarArrayLength(env, array) \
    (((*(env))->GetScalarArrayLength)(env, array))

#define JRI_GetScalarArrayData(env, array) \
    (((*(env))->GetScalarArrayData)(env, array))

/*****
 * Specific Scalar Array Types
 *****/

#define JRI_NewBooleanArray(env, length, initialValues) \
    JRI_NewScalarArray(env, length, (jbyte*)(initialValues))
#define JRI_GetBooleanArrayLength(env, array) \
    JRI_GetScalarArrayLength(env, array)
#define JRI_GetBooleanArrayData(env, array) \
    ((jbool*)JRI_GetScalarArrayData(env, array))

#define JRI_NewByteArray(env, length, initialValues) \
    JRI_NewScalarArray(env, length, (jbyte*)(initialValues))
#define JRI_GetByteArrayLength(env, array) \
    JRI_GetScalarArrayLength(env, array)
#define JRI_GetByteArrayData(env, array) \
    JRI_GetScalarArrayData(env, array)

#define JRI_NewCharArray(env, length, initialValues) \
    JRI_NewScalarArray(env, ((length) * sizeof(jchar)), \
    (jbyte*)(initialValues))
#define JRI_GetCharArrayLength(env, array) \
    JRI_GetScalarArrayLength(env, array)

```

```

#define JRI_GetCharArrayData(env, array) \
    ((jchar*)JRI_GetScalarArrayData(env, array))

#define JRI_NewShortArray(env, length, initialValues) \
    JRI_NewScalarArray(env, ((length) * sizeof(jshort)), \
    (jbyte*)(initialValues))
#define JRI_GetShortArrayLength(env, array) \
    JRI_GetScalarArrayLength(env, array)
#define JRI_GetShortArrayData(env, array) \
    ((jshort*)JRI_GetScalarArrayData(env, array))

#define JRI_NewIntArray(env, length, initialValues) \
    JRI_NewScalarArray(env, ((length) * sizeof(jint)), \
    (jbyte*)(initialValues))
#define JRI_GetIntArrayLength(env, array) \
    JRI_GetScalarArrayLength(env, array)
#define JRI_GetIntArrayData(env, array) \
    ((jint*)JRI_GetScalarArrayData(env, array))

#define JRI_NewLongArray(env, length, initialValues) \
    JRI_NewScalarArray(env, ((length) * sizeof(jlong)), \
    (jbyte*)(initialValues))
#define JRI_GetLongArrayLength(env, array) \
    JRI_GetScalarArrayLength(env, array)
#define JRI_GetLongArrayData(env, array) \
    ((jlong*)JRI_GetScalarArrayData(env, array))

#define JRI_NewFloatArray(env, length, initialValues) \
    JRI_NewScalarArray(env, ((length) * sizeof(jfloat)), \
    (jbyte*)(initialValues))
#define JRI_GetFloatArrayLength(env, array) \
    JRI_GetScalarArrayLength(env, array)
#define JRI_GetFloatArrayData(env, array) \
    ((jfloat*)JRI_GetScalarArrayData(env, array))

#define JRI_NewDoubleArray(env, length, initialValues) \
    JRI_NewScalarArray(env, ((length) * sizeof(jdouble)), \
    (jbyte*)(initialValues))
#define JRI_GetDoubleArrayLength(env, array) \
    JRI_GetScalarArrayLength(env, array)
#define JRI_GetDoubleArrayData(env, array) \
    ((jdouble*)JRI_GetScalarArrayData(env, array))

/*****
 * Object Array Operations
 *****/

#define JRI_NewObjectArray(env, length, elementClass, initialElement) \
    (((*(env))->NewObjectArray)(env, length, elementClass, initialElement))

#define JRI_GetObjectArrayLength(env, array) \
    (((*(env))->GetObjectArrayLength)(env, array))

#define JRI_GetObjectArrayElement(env, array, index) \
    (((*(env))->GetObjectArrayElement)(env, array, index))

#define JRI_SetObjectArrayElement(env, array, index, value) \

```

```

        (((*(env))->SetObjectArrayElement)(env, array, index, value))

/*****
 * Native Bootstrap
 *****/

/*
** This routine is used by the generated stub code to look up field
** positions and method identifiers for native access and invocation.
*/
#define JRI_RegisterNatives(env, clazz, nativeInfoArray) \
        (((*(env))->RegisterNatives)(env, clazz, nativeInfoArray))

#define JRI_UnregisterNatives(env, clazz) \
        (((*(env))->UnregisterNatives)(env, clazz))

/*****
 * Glue Routines -- These are used to auto-initialize fields
 *****/

JRI_PUBLIC_API(jref)
JRI_GetFieldByName(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jbool)
JRI_GetFieldByName_boolean(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jbyte)
JRI_GetFieldByName_byte(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jchar)
JRI_GetFieldByName_char(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jshort)
JRI_GetFieldByName_short(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jint)
JRI_GetFieldByName_int(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jlong)
JRI_GetFieldByName_long(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jfloat)
JRI_GetFieldByName_float(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jdouble)
JRI_GetFieldByName_double(JRIEnv* env, jref obj, JRIFieldThunk* field);

/*****

JRI_PUBLIC_API(void)
JRI_SetFieldByName(JRIEnv* env, jref obj, JRIFieldThunk* field, jref value);

JRI_PUBLIC_API(void)
JRI_SetFieldByName_boolean(JRIEnv* env, jref obj, JRIFieldThunk* field, jbool
value);

JRI_PUBLIC_API(void)

```



```

JRI_SetFieldByName_byte(JRIEnv* env, jref obj, JRIFieldThunk* field, jbyte
value);

JRI_PUBLIC_API(void)
JRI_SetFieldByName_char(JRIEnv* env, jref obj, JRIFieldThunk* field, jchar
value);

JRI_PUBLIC_API(void)
JRI_SetFieldByName_short(JRIEnv* env, jref obj, JRIFieldThunk* field, jshort
value);

JRI_PUBLIC_API(void)
JRI_SetFieldByName_int(JRIEnv* env, jref obj, JRIFieldThunk* field, jint value);

JRI_PUBLIC_API(void)
JRI_SetFieldByName_long(JRIEnv* env, jref obj, JRIFieldThunk* field, jlong
value);

JRI_PUBLIC_API(void)
JRI_SetFieldByName_float(JRIEnv* env, jref obj, JRIFieldThunk* field, jfloat
value);

JRI_PUBLIC_API(void)
JRI_SetFieldByName_double(JRIEnv* env, jref obj, JRIFieldThunk* field, jdouble
value);

/*****/

JRI_PUBLIC_API(jref)
JRI_GetStaticFieldByName(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jbool)
JRI_GetStaticFieldByName_boolean(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jbyte)
JRI_GetStaticFieldByName_byte(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jchar)
JRI_GetStaticFieldByName_char(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jshort)
JRI_GetStaticFieldByName_short(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jint)
JRI_GetStaticFieldByName_int(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jlong)
JRI_GetStaticFieldByName_long(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jfloat)
JRI_GetStaticFieldByName_float(JRIEnv* env, jref obj, JRIFieldThunk* field);

JRI_PUBLIC_API(jdouble)
JRI_GetStaticFieldByName_double(JRIEnv* env, jref obj, JRIFieldThunk* field);

/*****/

```

```

JRI_PUBLIC_API(void)
JRI_SetStaticFieldByName(JRIEnv* env, jref obj, JRIFieldThunk* field, jref
value);

JRI_PUBLIC_API(void)
JRI_SetStaticFieldByName_boolean(JRIEnv* env, jref obj, JRIFieldThunk* field,
jbool value);

JRI_PUBLIC_API(void)
JRI_SetStaticFieldByName_byte(JRIEnv* env, jref obj, JRIFieldThunk* field, jbyte
value);

JRI_PUBLIC_API(void)
JRI_SetStaticFieldByName_char(JRIEnv* env, jref obj, JRIFieldThunk* field, jchar
value);

JRI_PUBLIC_API(void)
JRI_SetStaticFieldByName_short(JRIEnv* env, jref obj, JRIFieldThunk* field,
jshort value);

JRI_PUBLIC_API(void)
JRI_SetStaticFieldByName_int(JRIEnv* env, jref obj, JRIFieldThunk* field, jint
value);

JRI_PUBLIC_API(void)
JRI_SetStaticFieldByName_long(JRIEnv* env, jref obj, JRIFieldThunk* field, jlong
value);

JRI_PUBLIC_API(void)
JRI_SetStaticFieldByName_float(JRIEnv* env, jref obj, JRIFieldThunk* field,
jfloat value);

JRI_PUBLIC_API(void)
JRI_SetStaticFieldByName_double(JRIEnv* env, jref obj, JRIFieldThunk* field,
jdouble value);

/*****
#ifdef __cplusplus
}
#endif
#endif /* JRI_H */
*****/

```

```

/* -*- Mode: C; tab-width: 4; -*- */
/*****
 * Java Runtime Interface - Machine Dependent Types
 * Copyright (c) 1996 Netscape Communications Corporation. All rights reserved.
 *****/

#ifndef JRI_MD_H
#define JRI_MD_H

#ifdef __cplusplus
extern "C" {
#endif

/*****

/* DLL Entry modifiers... */

#ifdef XP_PC
#define EXPORTED extern __declspec(dllexport)
#define JRI_PUBLIC_API(ResultType) __declspec(dllexport) ResultType
#define JRI_CALLBACK
#else /* !_WIN32 */
#define JRI_PUBLIC_API(ResultType) ResultType _cdecl _export _loadds
#define JRI_CALLBACK _loadds
#endif
#else /* !XP_PC */
#define JRI_PUBLIC_API(ResultType) ResultType
#define JRI_CALLBACK
#endif /* XP_PC */

#ifndef FAR /* for non-Win16 */
#define FAR
#endif

/*****

/* Java Scalar Types */

typedef unsigned char jbool;
typedef char jbyte;
typedef short jchar;
typedef short jshort;
#ifdef IS_64 /* XXX ok for alpha, but not right on all 64-bit architectures */
typedef unsigned int jint;
typedef int jint;
#else
typedef unsigned long jint;
typedef long jint;
#endif
typedef float jfloat;
typedef double jdouble;

typedef jint jsize;

/*****
 * jlong : long long (64-bit signed integer type) support.
 *****/

```

```

*****/

/*
** Bit masking macros.  (n must be <= 31 to be portable)
*/
#define JRI_BIT(n)                ((juint)1 << (n))
#define JRI_BITMASK(n)            (JRI_BIT(n) - 1)

#ifdef HAVE_LONG_LONG

#ifndef _WIN32
typedef long long                jlong;
typedef unsigned long long       julong;

#define jlong_MAXINT              0x7fffffffffffffffLL
#define jlong_MININT              0x8000000000000000LL
#define jlong_ZERO                0x0LL

#else
typedef LONGLONG                 jlong;
typedef DWORDLONG               julong;

#define jlong_MAXINT              0x7fffffffffffffffLL
#define jlong_MININT              0x8000000000000000LL
#define jlong_ZERO                0x0LL

#endif

#define jlong_IS_ZERO(a)          ((a) == 0)
#define jlong_EQ(a, b)            ((a) == (b))
#define jlong_NE(a, b)            ((a) != (b))
#define jlong_GE_ZERO(a)          ((a) >= 0)
#define jlong_CMP(a, op, b)       ((a) op (b))

#define jlong_AND(r, a, b)        ((r) = (a) & (b))
#define jlong_OR(r, a, b)         ((r) = (a) | (b))
#define jlong_XOR(r, a, b)        ((r) = (a) ^ (b))
#define jlong_OR2(r, a)           ((r) = (r) | (a))
#define jlong_NOT(r, a)           ((r) = ~(a))

#define jlong_NEG(r, a)            ((r) = -(a))
#define jlong_ADD(r, a, b)         ((r) = (a) + (b))
#define jlong_SUB(r, a, b)         ((r) = (a) - (b))

#define jlong_MUL(r, a, b)         ((r) = (a) * (b))
#define jlong_DIV(r, a, b)         ((r) = (a) / (b))
#define jlong_MOD(r, a, b)         ((r) = (a) % (b))

#define jlong_SHL(r, a, b)         ((r) = (a) << (b))
#define jlong_SHR(r, a, b)         ((r) = (a) >> (b))
#define jlong_USHR(r, a, b)        ((r) = (julong)(a) >> (b))
#define jlong_ISHL(r, a, b)        ((r) = ((jlong)(a)) << (b))

#define jlong_L2I(i, l)           ((i) = (int)(l))
#define jlong_L2UI(ui, l)         ((ui) = (unsigned int)(l))
#define jlong_L2F(f, l)           ((f) = (l))
#define jlong_L2D(d, l)           ((d) = (l))

```

```

#define jlong_I2L(l, i)      ((l) = (i))
#define jlong_UI2L(l, ui)    ((l) = (ui))
#define jlong_F2L(l, f)      ((l) = (f))
#define jlong_D2L(l, d)      ((l) = (d))

#define jlong_UDIVMOD(qp, rp, a, b) \
    (*(qp) = ((julong)(a) / (b)), \
    *(rp) = ((julong)(a) % (b)))

#else /* !HAVE_LONG_LONG */

typedef struct {
#ifdef IS_LITTLE_ENDIAN
    jint lo, hi;
#else
    jint hi, lo;
#endif
} jlong;
typedef jlong                julong;

extern jlong jlong_MAXINT, jlong_MININT, jlong_ZERO;

#define jlong_IS_ZERO(a)      (((a).hi == 0) && ((a).lo == 0))
#define jlong_EQ(a, b)       (((a).hi == (b).hi) && ((a).lo == (b).lo))
#define jlong_NE(a, b)       (((a).hi != (b).hi) || ((a).lo != (b).lo))
#define jlong_GE_ZERO(a)     (((a).hi >> 31) == 0)

/*
 * NB: jlong_CMP and jlong_UCMP work only for strict relationals (<, >).
 */
#define jlong_CMP(a, op, b)   (((int32)(a).hi op (int32)(b).hi) || \
                                (((a).hi == (b).hi) && ((a).lo op (b).lo)))
#define jlong_UCMP(a, op, b) (((a).hi op (b).hi) || \
                                (((a).hi == (b).hi) && ((a).lo op (b).lo)))

#define jlong_AND(r, a, b)    ((r).lo = (a).lo & (b).lo, \
                                (r).hi = (a).hi & (b).hi)
#define jlong_OR(r, a, b)     ((r).lo = (a).lo | (b).lo, \
                                (r).hi = (a).hi | (b).hi)
#define jlong_XOR(r, a, b)    ((r).lo = (a).lo ^ (b).lo, \
                                (r).hi = (a).hi ^ (b).hi)
#define jlong_OR2(r, a)       ((r).lo = (r).lo | (a).lo, \
                                (r).hi = (r).hi | (a).hi)
#define jlong_NOT(r, a)       ((r).lo = ~(a).lo, \
                                \
                                (r).hi = ~(a).hi)

#define jlong_NEG(r, a)        ((r).lo = -(int32)(a).lo, \
                                (r).hi = -(int32)(a).hi - ((r).lo != 0))
#define jlong_ADD(r, a, b) { \
    jlong _a, _b; \
    _a = a; _b = b; \
    (r).lo = _a.lo + _b.lo; \
    (r).hi = _a.hi + _b.hi + ((r).lo < _b.lo); \
}

```

```

#define jlong_SUB(r, a, b) {
    jlong _a, _b;
    _a = a; _b = b;
    (r).lo = _a.lo - _b.lo;
    (r).hi = _a.hi - _b.hi - (_a.lo < _b.lo);
}

/*
 * Multiply 64-bit operands a and b to get 64-bit result r.
 * First multiply the low 32 bits of a and b to get a 64-bit result in r.
 * Then add the outer and inner products to r.hi.
 */
#define jlong_MUL(r, a, b) {
    jlong _a, _b;
    _a = a; _b = b;
    jlong_MUL32(r, _a.lo, _b.lo);
    (r).hi += _a.hi * _b.lo + _a.lo * _b.hi;
}

/* XXX jlong_lo16(a) = ((a) << 16 >> 16) is better on some archs (not on mips)
 */
#define jlong_lo16(a)      ((a) & JRI_BITMASK(16))
#define jlong_hi16(a)      ((a) >> 16)

/*
 * Multiply 32-bit operands a and b to get 64-bit result r.
 * Use polynomial expansion based on primitive field element (1 << 16).
 */
#define jlong_MUL32(r, a, b) {
    jint _a1, _a0, _b1, _b0, _y0, _y1, _y2, _y3;
    _a1 = jlong_hi16(a), _a0 = jlong_lo16(a);
    _b1 = jlong_hi16(b), _b0 = jlong_lo16(b);
    _y0 = _a0 * _b0;
    _y1 = _a0 * _b1;
    _y2 = _a1 * _b0;
    _y3 = _a1 * _b1;
    _y1 += jlong_hi16(_y0);
    _y1 += _y2;
    if (_y1 < _y2) _y3 += 1 << 16;
    (r).lo = (jlong_lo16(_y1) << 16) + jlong_lo16(_y0);
    (r).hi = _y3 + jlong_hi16(_y1);
}

/*
 * Divide 64-bit unsigned operand a by 64-bit unsigned operand b, setting *qp
 * to the 64-bit unsigned quotient, and *rp to the 64-bit unsigned remainder.
 * Minimize effort if one of qp and rp is null.
 */
#define jlong_UDIVMOD(qp, rp, a, b) jlong_udivmod(qp, rp, a, b)

extern JRI_PUBLIC_API(void)
jlong_udivmod(julong *qp, julong *rp, julong a, julong b);

#define jlong_DIV(r, a, b) {
    jlong _a, _b;
    jint _negative = (int32)(a).hi < 0;
    if (_negative) {

```

```

    jlong_NEG(_a, a);
} else {
    _a = a;
}
if ((int32)(b).hi < 0) {
    _negative ^= 1;
    jlong_NEG(_b, b);
} else {
    _b = b;
}
jlong_UDIVMOD(&(r), 0, _a, _b);
if (_negative)
    jlong_NEG(r, r);
}

#define jlong_MOD(r, a, b) {
    jlong _a, _b;
    jint _negative = (int32)(a).hi < 0;
    if (_negative) {
        jlong_NEG(_a, a);
    } else {
        _a = a;
    }
    if ((int32)(b).hi < 0) {
        jlong_NEG(_b, b);
    } else {
        _b = b;
    }
    jlong_UDIVMOD(0, &(r), _a, _b);
    if (_negative)
        jlong_NEG(r, r);
}

/*
 * NB: b is a jint, not jlong or julong, for the shift ops.
 */
#define jlong_SHL(r, a, b) {
    if (b) {
        jlong _a;
        _a = a;
        if ((b) < 32) {
            (r).lo = _a.lo << (b);
            (r).hi = (_a.hi << (b)) | (_a.lo >> (32 - (b)));
        } else {
            (r).lo = 0;
            (r).hi = _a.lo << ((b) & 31);
        }
    } else {
        (r) = (a);
    }
}

/* a is an int32, b is int32, r is jlong */
#define jlong_ISHL(r, a, b) {
    if (b) {
        jlong _a;
        _a.lo = (a);

```

```

    _a.hi = 0;
    if ((b) < 32) {
        (r).lo = (a) << (b);
        (r).hi = ((a) >> (32 - (b)));
    } else {
        (r).lo = 0;
        (r).hi = (a) << ((b) & 31);
    }
} else {
    (r).lo = (a);
    (r).hi = 0;
}
}

#define jlong_SHR(r, a, b) {
    if (b) {
        jlong _a;
        _a = a;
        if ((b) < 32) {
            (r).lo = (_a.hi << (32 - (b))) | (_a.lo >> (b));
            (r).hi = (int32)_a.hi >> (b);
        } else {
            (r).lo = (int32)_a.hi >> ((b) & 31);
            (r).hi = (int32)_a.hi >> 31;
        }
    } else {
        (r) = (a);
    }
}

#define jlong_USHR(r, a, b) {
    if (b) {
        jlong _a;
        _a = a;
        if ((b) < 32) {
            (r).lo = (_a.hi << (32 - (b))) | (_a.lo >> (b));
            (r).hi = _a.hi >> (b);
        } else {
            (r).lo = _a.hi >> ((b) & 31);
            (r).hi = 0;
        }
    } else {
        (r) = (a);
    }
}

#define jlong_L2I(i, l)      ((i) = (l).lo)
#define jlong_L2UI(ui, l)   ((ui) = (l).lo)
#define jlong_L2F(f, l)     { double _d; jlong_L2D(_d, l); (f) = (float) _d; }

#define jlong_L2D(d, l) {
    int32 _negative;
    jlong _absval;

    _negative = (l).hi >> 31;
    if (_negative) {
        jlong_NEG(_absval, l);
    }
}

```



```

    } else {
        _absval = 1;
    }
    (d) = (double)_absval.hi * 4.294967296e9 + _absval.lo;
    if (_negative)
        (d) = -(d);
}

#define jlong_I2L(l, i)      ((l).hi = (i) >> 31, (l).lo = (i))
#define jlong_UI2L(l, ui)   ((l).hi = 0, (l).lo = (ui))
#define jlong_F2L(l, f)     { double _d = (double) f; jlong_D2L(l, _d); }

#define jlong_D2L(l, d) {
    int _negative;
    double _absval, _d_hi;
    jlong _lo_d;

    _negative = ((d) < 0);
    _absval = _negative ? -(d) : (d);

    (l).hi = (juint)(_absval / 4.294967296e9);
    (l).lo = 0;
    jlong_L2D(_d_hi, l);
    _absval -= _d_hi;
    _lo_d.hi = 0;
    if (_absval < 0) {
        _lo_d.lo = (juint) -_absval;
        jlong_SUB(l, l, _lo_d);
    } else {
        _lo_d.lo = (juint) _absval;
        jlong_ADD(l, l, _lo_d);
    }

    if (_negative)
        jlong_NEG(l, l);
}

#endif /* !HAVE_LONG_LONG */

/*****
**
** JDK Stuff -- This stuff is still needed while we're using the JDK
** dynamic linking strategy to call native methods.
**
*****/

typedef union JRI_JDK_stack_item {
    /* Non pointer items */
    jint      i;
    jfloat    f;
    jint      o;
    /* Pointer items */
    void      *h;
    void      *p;
    unsigned char *addr;
#ifdef OSF1
    double    d;
    long      l;          /* == 64bits! */

```

```

#endif
} JRI_JDK_stack_item;

typedef union JRI_JDK_Java8Str {
    jint x[2];
    jdouble d;
    jlong l;
    void *p;
    float f;
} JRI_JDK_Java8;

#ifdef HAVE_ALIGNED_LONGLONGS
#define JRI_GET_INT64(_t, _addr) ( ((_t).x[0] = ((jint*)(_addr))[0]), \
                                   ((_t).x[1] = ((jint*)(_addr))[1]), \
                                   (_t).l )
#define JRI_SET_INT64(_t, _addr, _v) ( (_t).l = (_v), \
                                         ((jint*)(_addr))[0] = (_t).x[0], \
                                         ((jint*)(_addr))[1] = (_t).x[1] )
#else
#define JRI_GET_INT64(_t, _addr) (*(jlong*)(_addr))
#define JRI_SET_INT64(_t, _addr, _v) (*(jlong*)(_addr) = (_v))
#endif

/* If double's must be aligned on doubleword boundaries then define this */
#ifdef HAVE_ALIGNED_DOUBLES
#define JRI_GET_DOUBLE(_t, _addr) ( ((_t).x[0] = ((jint*)(_addr))[0]), \
                                   ((_t).x[1] = ((jint*)(_addr))[1]), \
                                   (_t).d )
#define JRI_SET_DOUBLE(_t, _addr, _v) ( (_t).d = (_v), \
                                         ((jint*)(_addr))[0] = (_t).x[0], \
                                         ((jint*)(_addr))[1] = (_t).x[1] )
#else
#define JRI_GET_DOUBLE(_t, _addr) (*(jdouble*)(_addr))
#define JRI_SET_DOUBLE(_t, _addr, _v) (*(jdouble*)(_addr) = (_v))
#endif

/*****
#ifdef __cplusplus
}
#endif
#endif /* JRI_MD_H */
*****/

```

```

# Generated automatically from Makefile.in by configure.
PACKAGE          = plugext
VERSION          = 1.0
OBJS             = plugext.o plugextCmd.o plugextCmdAZ.o NPN_wrapper.o plugInstCmd.o
plugInstCmdAZ.o extensionLib.o

#-----
prefix           = /usr/local
exec_prefix     = ${prefix}

TCL_VERSION     = 8.0

INSTALL          = /usr/bin/install -c
INSTALL_PROGRAM = ${INSTALL}
INSTALL_DATA     = ${INSTALL} -m 644
SHLIB_CFLAGS     = -fPIC
SHLIB_SUFFIX    = .so
SHLIB_LD        = cc -shared
## HP cc sometimes requires -Aa for proper ansi compilation
TCL_CFLAGS      = -DXP_UNIX -g

LIB_RUNTIME_DIR = ${exec_prefix}/lib

INCLUDES        = -I${prefix}/include -Iinclude -I../Netscape/include

DLL             = $(PACKAGE)${SHLIB_SUFFIX}
DLLDIR          = /usr/local/lib/${PACKAGE}${VERSION}

CSWITCHES       = -I. $(TCL_CFLAGS) $(SHLIB_CFLAGS) $(INCLUDES)

all: Makefile $(DLL) pkgIndex.tcl dlopen.so

# Makefile: Makefile.in config.status
#      $(SHELL) config.status

config.status: configure
      $(SHELL) config.status --recheck

$(DLL): $(OBJS) plugAncill.so
      $(SHLIB_LD) -o $@ $(OBJS)

plugAncill.so: plugAncill.o
      $(SHLIB_LD) -o $@ plugAncill.o

dlopen.so: dlopen.o
      $(SHLIB_LD) -o $@ dlopen.o

plugext.o: plugext.c
      $(CC) $(CSWITCHES) $(CPPFLAGS) -c plugext.c

plugextCmd.o: plugextCmd.c
      $(CC) $(CSWITCHES) $(CPPFLAGS) -c plugextCmd.c

plugextCmdAZ.o: plugextCmdAZ.c
      $(CC) $(CSWITCHES) $(CPPFLAGS) -c plugextCmdAZ.c

NPN_wrapper.o: NPN_wrapper.c

```

```

$(CC) $(CSWITCHES) $(CPPFLAGS) -c NPN_wrapper.c

plugAncill.o: plugext.c
$(CC) $(CSWITCHES) $(CPPFLAGS) -c plugAncill.c

plugInstCmd.o: plugInstCmd.c
$(CC) $(CSWITCHES) $(CPPFLAGS) -c plugInstCmd.c

plugInstCmdAZ.o: plugInstCmdAZ.c
$(CC) $(CSWITCHES) $(CPPFLAGS) -c plugInstCmdAZ.c

extensionLib.o: extensionLib.c
$(CC) $(CSWITCHES) $(CPPFLAGS) -c extensionLib.c

pkgIndex.tcl:
(\
echo 'if {[catch {package require Tcl $(TCL_VERSION)}}} return';\
echo 'package ifneeded $(PACKAGE) $(VERSION)\
[list load [file join $$dir $(DLL).$(VERSION)] $(PACKAGE)]'\
) > pkgIndex.tcl

install: all
if test ! -d "$(DLLDIR)"; then mkdir "$(DLLDIR)"; fi
$(INSTALL_PROGRAM) $(DLL) "$(DLLDIR)/$(DLL).$(VERSION)"
$(INSTALL_DATA) pkgIndex.tcl "$(DLLDIR)/pkgIndex.tcl"

test: all
tclsh$(TCL_VERSION) testplugext.tcl

clean:
rm -f *.o *$(SHLIB_SUFFIX) core pkgIndex.tcl test.*

distclean: clean
rm -f Makefile config.cache config.log config.status

```

```

/* -*- Mode: C; tab-width: 4; -*- */
/*
 * npapi.h $Revision: 1.1 $
 * Netscape client plug-in API spec
 */

```

```

#ifndef _NPAPI_H_
#define _NPAPI_H_

```

```

#include "jri.h"          /* Java Runtime Interface */

```

```

/* XXX this needs to get out of here */
#if defined(__MWERKS__)
#ifndef XP_MAC
#define XP_MAC
#endif
#endif

```

```

/*-----*/
/*          Plugin Version Constants          */
/*-----*/

```

```

#define NP_VERSION_MAJOR 0
#define NP_VERSION_MINOR 9

```

```

/*-----*/
/*          Definition of Basic Types          */
/*-----*/

```

```

#ifndef _UINT16
typedef unsigned short uint16;
#endif
#ifndef _UINT32
#if defined(__alpha)
typedef unsigned int uint32;
#else /* __alpha */
typedef unsigned long uint32;
#endif /* __alpha */
#endif
#ifndef _INT16
typedef short int16;
#endif
#ifndef _INT32
#if defined(__alpha)
typedef int int32;
#else /* __alpha */
typedef long int32;
#endif /* __alpha */
#endif

```

```

#ifndef FALSE
#define FALSE (0)

```

```

#endif
#ifndef TRUE
#define TRUE (1)
#endif
#ifndef NULL
#define NULL (0L)
#endif

typedef unsigned char  NPBool;
typedef void*          NPEvent;
typedef int16          NPError;
typedef int16          NPReason;
typedef char*          NPMIMEType;

/*-----*/
/*                Structures and definitions                */
/*-----*/

/*
 * NPP is a plug-in's opaque instance handle
 */
typedef struct _NPP
{
    void*  pdata;          /* plug-in private data */
    void*  ndata;          /* netscape private data */
} NPP_t;

typedef NPP_t*  NPP;

typedef struct _NPStream
{
    void*      pdata;          /* plug-in private data */
    void*      ndata;          /* netscape private data */
    const char* url;
    uint32     end;
    uint32     lastmodified;
    void*      notifyData;
} NPStream;

typedef struct _NPByteRange
{
    int32     offset;          /* negative offset means from the end */
    uint32     length;
    struct _NPByteRange* next;
} NPByteRange;

typedef struct _NPSavedData
{
    int32     len;
    void*     buf;
} NPSavedData;

```

```

typedef struct _NPRect
{
    uint16  top;
    uint16  left;
    uint16  bottom;
    uint16  right;
} NPRect;

#ifdef XP_UNIX
/*
 * Unix specific structures and definitions
 */
#include <X11/Xlib.h>

/*
 * Callback Structures.
 *
 * These are used to pass additional platform specific information.
 */
enum {
    NP_SETWINDOW = 1
};

typedef struct
{
    int32 type;
} NPAnyCallbackStruct;

typedef struct
{
    int32          type;
    Display*       display;
    Visual*        visual;
    Colormap       colormap;
    unsigned int   depth;
} NPSetWindowCallbackStruct;

/*
 * List of variable names for which NPP_GetValue shall be implemented
 */
typedef enum {
    NPPVpluginNameString = 1,
    NPPVpluginDescriptionString
} NPPVariable;

/*
 * List of variable names for which NPN_GetValue is implemented by Mozilla
 */
typedef enum {
    NPNVxDisplay = 1,
    NPNVxtAppContext
} NPNVariable;

#endif /* XP_UNIX */

```

```

typedef struct _NPWindow
{
    void*    window;           /* Platform specific window handle */
    uint32 x;                  /* Position of top left corner relative */
    uint32 y;                  /* to a netscape page. */
    uint32 width;              /* Maximum window size */
    uint32 height;
    NPRect clipRect;          /* Clipping rectangle in port coordinates */
                                /* Used by MAC only. */
#ifdef XP_UNIX
    void*    ws_info;          /* Platform-dependent additional data */
#endif /* XP_UNIX */
} NPWindow;

```

```

typedef struct _NPFullPrint
{
    NPBool pluginPrinted;      /* Set TRUE if plugin handled fullscreen */
                                /* printing */
    NPBool printOne;           /* TRUE if plugin should print one copy */
                                /* to default printer */
    void* platformPrint;       /* Platform-specific printing info */
} NPFullPrint;

```

```

typedef struct _NPEmbedPrint
{
    NPWindow window;
    void* platformPrint;       /* Platform-specific printing info */
} NPEmbedPrint;

```

```

typedef struct _NPPrint
{
    uint16 mode;               /* NP_FULL or NP_EMBED */
    union
    {
        NPFullPrint fullPrint; /* if mode is NP_FULL */
        NPEmbedPrint embedPrint; /* if mode is NP_EMBED */
    } print;
} NPPrint;

```

```

#ifdef XP_MAC
/*
 * Mac-specific structures and definitions.
 */

```

```

#include <Quickdraw.h>
#include <Events.h>

```

```

typedef struct NP_Port
{
    CGrafPtr port;            /* Grafport */
    int32 portx;              /* position inside the topmost window */
    int32 porty;

```



```

} NP_Port;

/*
 * Non-standard event types that can be passed to HandleEvent
 */
#define getFocusEvent      (osEvt + 16)
#define loseFocusEvent     (osEvt + 17)
#define adjustCursorEvent  (osEvt + 18)

#endif /* XP_MAC */

/*
 * Values for mode passed to NPP_New:
 */
#define NP_EMBED            1
#define NP_FULL             2

/*
 * Values for stream type passed to NPP_NewStream:
 */
#define NP_NORMAL           1
#define NP_SEEK             2
#define NP_ASFILE           3
#define NP_ASFILEONLY       4

#define NP_MAXREADY         (((unsigned) (~0) << 1) >> 1)

/*-----*/
/*                               Error and Reason Code definitions                               */
/*-----*/

/*
 * Values of type NPErrors:
 */
#define NPERR_BASE          0
#define NPERR_NO_ERROR      (NPERR_BASE + 0)
#define NPERR_GENERIC_ERROR (NPERR_BASE + 1)
#define NPERR_INVALID_INSTANCE_ERROR (NPERR_BASE + 2)
#define NPERR_INVALID_FUNCTABLE_ERROR (NPERR_BASE + 3)
#define NPERR_MODULE_LOAD_FAILED_ERROR (NPERR_BASE + 4)
#define NPERR_OUT_OF_MEMORY_ERROR (NPERR_BASE + 5)
#define NPERR_INVALID_PLUGIN_ERROR (NPERR_BASE + 6)
#define NPERR_INVALID_PLUGIN_DIR_ERROR (NPERR_BASE + 7)
#define NPERR_INCOMPATIBLE_VERSION_ERROR (NPERR_BASE + 8)
#define NPERR_INVALID_PARAM (NPERR_BASE + 9)
#define NPERR_INVALID_URL   (NPERR_BASE + 10)
#define NPERR_FILE_NOT_FOUND (NPERR_BASE + 11)
#define NPERR_NO_DATA        (NPERR_BASE + 12)
#define NPERR_STREAM_NOT_SEEKABLE (NPERR_BASE + 13)

/*
 * Values of type NPReason:
 */
#define NPRES_BASE          0

```

```

#define NPRES_NETWORK_ERR (NPRES_BASE + 0)
#define NPRES_USER_BREAK (NPRES_BASE + 2)
#define NPRES_DONE (NPRES_BASE + 3)

/*
 * Don't use these obsolete error codes any more.
 */
#define NP_NOERR NP_NOERR_is_obsolete_use_NPERR_NO_ERROR
#define NP_EINVAL NP_EINVAL_is_obsolete_use_NPERR_GENERIC_ERROR
#define NP_EABORT NP_EABORT_is_obsolete_use_NPRES_USER_BREAK

/*-----*/
/* Function Prototypes */
/*-----*/

#if defined(_WINDOWS) && !defined(WIN32)
#define NP_LOADDDS _loadds
#else
#define NP_LOADDDS
#endif

#ifdef __cplusplus
extern "C" {
#endif

/*
 * NPP_* functions are provided by the plugin and called by the navigator.
 */

#ifdef XP_UNIX
char* NPP_GetMIMEDescription(void);
NPError NPP_GetValue(void *, NPPVariable variable,
void *value);
#endif /* XP_UNIX */
NPError NPP_Initialize(void);
void NPP_Shutdown(void);
NPError NP_LOADDDS NPP_New(NPMIMETYPE pluginType, NPP instance,
uint16 mode, int16 argc, char* argn[],
char* argv[], NPSavedData* saved);
NPError NP_LOADDDS NPP_Destroy(NPP instance, NPSavedData** save);
NPError NP_LOADDDS NPP_SetWindow(NPP instance, NPWindow* window);
NPError NP_LOADDDS NPP_NewStream(NPP instance, NPMIMETYPE type,
NPStream* stream, NPBool seekable,
uint16* stype);
NPError NP_LOADDDS NPP_DestroyStream(NPP instance, NPStream* stream,
NPReason reason);
int32 NP_LOADDDS NPP_WriteReady(NPP instance, NPStream* stream);
int32 NP_LOADDDS NPP_Write(NPP instance, NPStream* stream, int32 offset,
int32 len, void* buffer);
void NP_LOADDDS NPP_StreamAsFile(NPP instance, NPStream* stream,
const char* fname);
void NP_LOADDDS NPP_Print(NPP instance, NPPrint* platformPrint);
int16 NPP_HandleEvent(NPP instance, void* event);
void NPP_URLNotify(NPP instance, const char* url,

```

```

jref                                     NPReason reason, void* notifyData);
NPP_GetJavaClass(void);

/*
 * NPN_* functions are provided by the navigator and called by the plugin.
 */

#ifdef XP_UNIX
NPError                                     NPN_GetValue(NPP instance, NPNVariable
variable,                                     void *value);

#endif /* XP_UNIX */
void                                     NPN_Version(int* plugin_major, int* plugin_minor,
int* netscape_major, int*
netscape_minor);
NPError                                     NPN_GetURLNotify(NPP instance, const char* url,
const char* target, void* notifyData);
NPError                                     NPN_GetURL(NPP instance, const char* url,
const char* target);
NPError                                     NPN_PostURLNotify(NPP instance, const char* url,
const char* target, uint32 len, const
char* buf,
NPBool file, void* notifyData);
NPError                                     NPN_PostURL(NPP instance, const char* url,
const char* target, uint32 len, const
char* buf,
NPBool file);
NPError                                     NPN_RequestRead(NPStream* stream, NPByteRange*
rangeList);
NPError                                     NPN_NewStream(NPP instance, NPMIMEType type,
const char* target, NPStream**
stream);
int32                                     NPN_Write(NPP instance, NPStream* stream, int32
len,
void* buffer);
NPError                                     NPN_DestroyStream(NPP instance, NPStream* stream,
NPReason reason);
void                                     NPN_Status(NPP instance, const char* message);
const char*                               NPN_UserAgent(NPP instance);
void*                                     NPN_MemAlloc(uint32 size);
void                                     NPN_MemFree(void* ptr);
uint32                                    NPN_MemFlush(uint32 size);
void                                     NPN_ReloadPlugins(NPBool reloadPages);
JRIEnv*                                   NPN_GetJavaEnv(void);
jref                                       NPN_GetJavaPeer(NPP instance);

#ifdef __cplusplus
} /* end extern "C" */
#endif

#endif /* _NPAPI_H_ */

```

```

#if defined (_WIN32)
#include <windows.h>
#include <stdio.h>
#endif
#include <stdlib.h>
#include <string.h>
#include "npupp.h"
#include "plugextInt.h"
extern Tcl_HashTable *plugext_hashtablePtr;

#include <stdarg.h>

/* FOR DEBUGGING PURPOSES - takes a set of chars and makes a null terminated
   string
*/

static char tmpString[512];

FILE *logFile = NULL;

void MyOpenit () {
    logFile = fopen("NPNLog", "a");
}

void MyCloseIt () {
    fclose(logFile);
}

void MyPrint(char *fmt, ...) {
    va_list ap;

    MyOpenit();

    va_start(ap, fmt);

    fprintf(logFile, fmt, ap);
    MyCloseIt();
}

char * makeNullTerm(char *input, int len) {

    if (len > 511) len = 511;

    memcpy(tmpString, input, len);
    tmpString[len] = 0;
    return tmpString;
}

#ifdef XP_UNIX
NPErrors PlugGetValue(NPP NPPhandle, NPNVariable variable, void *value) {
    MyPrint( "--- PlugGetValue(NPP NPPhandle, NPNVariable variable, called %xx
%d %xx\n",
            NPPhandle, variable, value);
    fflush(stdout);
}

```

```

    }
#endif /* XP_UNIX */

void PlugVersion(int* plugin_major, int* plugin_minor,
    int* netscape_major, int* netscape_minor) {
    MyPrint( "---- PlugVersion(int* plugin_major, int* plugin_minor, called %d
%d\n",
        plugin_major, plugin_minor);
    fflush(stdout);
    *plugin_minor = 9;
    *plugin_major = 0;
}

NPError PlugGetURLNotify(NPP NPPhandle, const char* url, const char* target,
void* notifyData) {
    MyPrint( "---- PlugGetURLNotify(NPP NPPhandle, const char* url, called %xx
%s %s %xx\n",
        NPPhandle, url, target, notifyData);
    fflush(stdout);
    return (0);
}

NPError PlugGetURL(NPP NPPhandle, const char* url, const char* target) {
    MyPrint( "---- PlugGetURL(NPP NPPhandle, const char* url, called %xx %s
%s\n",
        NPPhandle, url, target);
    fflush(stdout);
    return (0);
}

NPError PlugPostURLNotify(NPP NPPhandle, const char* url,
    const char* target, uint32 len,
    const char* buf, NPBool file,
    void* notifyData) {
    MyPrint( "---- PlugPostURLNotify(NPP NPPhandle, const char* url, called %xx
%s %s %d %s %xx %xx\n",
        NPPhandle, url, target, len, buf, file, notifyData);
    fflush(stdout);
    return (0);
}

NPError PlugPostURL(NPP NPPhandle, const char* url,
    const char* target, uint32 len,
    const char* buf, NPBool file) {
    MyPrint( "---- PlugPostURL(NPP NPPhandle, const char* url, called %xx %s %s
%d %s %xx\n",
        NPPhandle, url, target, len, buf, file);
    fflush(stdout);
    return (0);
}

NPError PlugRequestRead(NPStream* stream, NPByteRange* rangeList) {
    MyPrint( "---- PlugRequestRead(NPStream* stream, NPByteRange* rangeList %xx
%xx called\n", stream, rangeList);
    fflush(stdout);
    return (0);
}

```

```

NPError PlugNewStream(NPP NPPhandle, NPMIMETYPE type,
                     const char* target, NPStream** stream) {
    NPStream *str;
    Tcl_HashEntry *hashEntryPtr;
    Tcl_HashSearch hashSearch;
    InstanceData *instanceData;

    hashEntryPtr = Tcl_FirstHashEntry(plugext_hashtablePtr, &hashSearch);
    instanceData = (InstanceData *) Tcl_GetHashValue(hashEntryPtr);

    while (instanceData->handle != NPPhandle) {
        hashEntryPtr = Tcl_NextHashEntry(&hashSearch);
        if (hashEntryPtr == NULL) {
            return -1;
        }
        instanceData = (InstanceData *) Tcl_GetHashValue(hashEntryPtr);
    }

    *stream = (NPStream *) Tcl_Alloc(sizeof (NPStream));
    str = *stream;
    MyPrint( "---PlugNewStream(NPP instance, %xx NPMIMETYPE type %s, target:
%s *stream: %xx INstance::pdata: %xx ndata: %xx\n",
    NPPhandle, type, target, stream, NPPhandle->pdata, NPPhandle->ndata);
    fflush(stdout);
    str->pdata = instanceData;
    str->url = "Test";
    MyPrint( "---PlugNewStream post *streams: %xx\n", str->pdata);
    fflush(stdout);
    return 0;
}

int32 PlugWrite(NPP NPPhandle, NPStream* stream, int32 len,
               void* buffer) {
    InstanceData *instanceData;
    instanceData = stream->pdata;
    if (instanceData->readBuffer == NULL) {
        instanceData->readBuffer = Tcl_Alloc(len*10);
        instanceData->readSize = len*10;
        instanceData->readOffset = 0;
        instanceData->readLen = 0;
    }

    if ((instanceData->readOffset + len) > instanceData->readSize) {
        instanceData->readBuffer = realloc(instanceData->readBuffer,
instanceData->readSize * 2);
        instanceData->readSize = instanceData->readSize * 2;
    }
    memcpy(instanceData->readBuffer + instanceData->readLen, buffer, len);
    instanceData->readLen += len ;

    MyPrint( "--- PlugWrite(NPP NPPhandle %xx, NPStream* stream %xx, int32 len
%xx, buf: %s called\n",
    NPPhandle, stream, len, makeNullTerm(buffer, len) );
    MyPrint( "--- PlugWrite stream->pdata: %xx", stream->pdata);
    fflush(stdout);
}

```

```

        return (0);
    }

NPError PlugDestroyStream(NPP NPPhandle, NPStream* stream,
                          NPReason reason) {
    MyPrint( "--- PlugDestroyStream(NPP NPPhandle, NPStream* stream,
called\n");
    fflush(stdout);
    return (0);
}

void PlugStatus(NPP NPPhandle, const char* message) {
    MyPrint( "--- PlugStatus(NPP NPPhandle, const char* message); called\n");
    MyPrint( "PlugStatus message: %s\n", message);
    fflush(stdout);
}

const char* PlugUserAgent(NPP NPPhandle) {
    MyPrint( "--- char* PlugUserAgent(NPP NPPhandle); called\n");
    fflush(stdout);
    return ("mozilla 3.0");
}

void* PlugMemAlloc(uint32 size) {
    MyPrint( "--- PlugMemAlloc(uint32 size) %d; called\n", size);
    fflush(stdout);
    return (Tcl_Alloc(size));
}

void PlugMemFree(void* ptr) {
    MyPrint( "--- PlugMemFree(void* ptr); called\n");
    fflush(stdout);
    Tcl_Free(ptr);
}

uint32 PlugMemFlush(uint32 size) {
    MyPrint( "--- PlugMemFlush(uint32 size) \n");
    fflush(stdout);
    return (0);
}

void PlugReloadPlugins(NPBool reloadPages){
    MyPrint( "--- PlugReloadPlugins(NPBool reloadPages)\n");
    fflush(stdout);
}

JRIEnv* PlugGetJavaEnv(void) {
    MyPrint( "--- PlugGetJavaEnv(void)\n");
    fflush(stdout);
    return (NULL);
}

jref PlugGetJavaPeer(NPP NPPhandle){
    MyPrint( "--- PlugGetJavaPeer(NPP NPPhandle)\n");
    fflush(stdout);
    return (NULL);
}

```

```

int setPlugextEntryPoints(NPNetscapeFuncs *n) {
MyPrint( "--- hit NP_GetEntryPoints: %d %xx\n", n->size, n); fflush(stdout);
MyPrint( "--- hit NP_GetEntryPoints: %xx\n", PlugUserAgent); fflush(stdout);
MyPrint( "--- hit NP_GetEntryPoints: %xx\n", PlugMemAlloc); fflush(stdout);
MyPrint( "--- hit NP_GetEntryPoints: %xx\n", PlugMemFlush); fflush(stdout);
MyPrint( "--- hit NP_GetEntryPoints: %xx\n", PlugMemFree); fflush(stdout);
    if (n->size < sizeof(NPNetscapeFuncs)) {
        return(NPERR_INVALID_FUNCTABLE_ERROR);
    }
    n->geturl = PlugGetURL ;
    n->posturl = PlugPostURL ;
    n->requestread = PlugRequestRead ;
    n->newstream = PlugNewStream ;
    n->write = PlugWrite ;
    n->destroystream = PlugDestroyStream ;
    n->status = PlugStatus ;
    n->uagent = PlugUserAgent ;
    n->memalloc = PlugMemAlloc ;
    n->memfree = PlugMemFree ;
    n->memflush = PlugMemFlush ;
    n->reloadplugins = PlugReloadPlugins ;
    n->getJavaEnv = PlugGetJavaEnv ;
    n->getJavaPeer = PlugGetJavaPeer ;
    n->geturlnotify = PlugGetURLNotify ;
    n->posturlnotify = PlugPostURLNotify ;
#ifdef XP_UNIX
    n->getvalue = PlugGetValue ;
#endif
}

```



```

/*
Forward declarations for NPN_wrapper.c
*/

#ifndef NPN_WRAPPER_H
#define NPN_WRAPPER_H

#ifndef NPUPP_H_
#include "npupp.h"
#endif

char *makeNullTerm(char *, int );
NPErrors PlugGetValue(NPP , void *, void *);
void PlugVersion(int*, int*, int*, int* );
NPErrors PlugGetURLNotify(NPP, const char*, const char*, void*);
NPErrors PlugGetURL(NPP, const char*, const char*);
NPErrors PlugPostURLNotify(NPP, const char*, const char*, uint32,
                           const char*, NPBool, void*);
NPErrors PlugPostURL(NPP, const char*, const char*, uint32,
                     const char*, NPBool);
NPErrors PlugRequestRead(NPStream*, NPByteRange*);
NPErrors PlugNewStream(NPP, NPMIMEType, const char*, NPStream**);
int32 PlugWrite(NPP, NPStream*, int32, void*);
NPErrors PlugDestroyStream(NPP, NPStream*, NPReason);
void PlugStatus(NPP, const char*);
const char* PlugUserAgent(NPP);
void* PlugMemAlloc(uint32);
void PlugMemFree(void*);
uint32 PlugMemFlush(uint32);
void PlugReloadPlugins(NPBool);
JRIEnv* PlugGetJavaEnv(void);
jref PlugGetJavaPeer(NPP);
int setPlugextEntryPoints(NPNetscapeFuncs *) ;

#endif

```

```

/* -*- Mode: C; tab-width: 4; -*- */
/*
 * npupp.h $Revision: 1.1 $
 * function call mechanics needed by platform specific glue code.
 */

#ifndef _NPUPP_H_
#define _NPUPP_H_

#ifndef GENERATINGCFM
#define GENERATINGCFM 0
#endif

#ifndef _NPAPI_H_
#include "npapi.h"
#endif

#include "jri.h"

/*****
*****
    plug-in function table macros
        for each function in and out of the plugin API we define
            typedef NPP_FooUPP
                #define NewNPP_FooProc
                #define CallNPP_FooProc
            for mac, define the UPP magic for PPC/68K calling
*****
*****/

/* NPP_Initialize */

#if GENERATINGCFM
typedef UniversalProcPtr NPP_InitializeUPP;

enum {
    uppNPP_InitializeProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(0))
        | RESULT_SIZE(SIZE_CODE(0))
};

#define NewNPP_InitializeProc(FUNC) \
    (NPP_InitializeUPP) NewRoutineDescriptor((ProcPtr) (FUNC), \
    uppNPP_InitializeProcInfo, GetCurrentArchitecture())
#define CallNPP_InitializeProc(FUNC) \
    (void) CallUniversalProc((UniversalProcPtr) (FUNC), \
    uppNPP_InitializeProcInfo)

#else

typedef void (*NPP_InitializeUPP)(void);
#define NewNPP_InitializeProc(FUNC) \
    ((NPP_InitializeUPP) (FUNC))
#define CallNPP_InitializeProc(FUNC) \

```

```

        (*(FUNC)) ()

#endif

/* NPP_Shutdown */

#if GENERATINGCFM
typedef UniversalProcPtr NPP_ShutdownUPP;

enum {
    uppNPP_ShutdownProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(0))
        | RESULT_SIZE(SIZE_CODE(0))
};

#define NewNPP_ShutdownProc(FUNC) \
    ((NPP_ShutdownUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPP_ShutdownProcInfo, GetCurrentArchitecture()))
#define CallNPP_ShutdownProc(FUNC) \
    (void)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPP_ShutdownProcInfo)

#else

typedef void (*NPP_ShutdownUPP)(void);
#define NewNPP_ShutdownProc(FUNC) \
    ((NPP_ShutdownUPP)(FUNC))
#define CallNPP_ShutdownProc(FUNC) \
    (*(FUNC)) ()

#endif

/* NPP_New */

#if GENERATINGCFM
typedef UniversalProcPtr NPP_NewUPP;

enum {
    uppNPP_NewProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPMIMType)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(uint16)))
        | STACK_ROUTINE_PARAMETER(4, SIZE_CODE(sizeof(int16)))
        | STACK_ROUTINE_PARAMETER(5, SIZE_CODE(sizeof(char **)))
        | STACK_ROUTINE_PARAMETER(6, SIZE_CODE(sizeof(char **)))
        | STACK_ROUTINE_PARAMETER(7, SIZE_CODE(sizeof(NPSavedData *)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPError)))
};

#define NewNPP_NewProc(FUNC) \
    ((NPP_NewUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPP_NewProcInfo, GetCurrentArchitecture()))
#define CallNPP_NewProc(FUNC, ARG1, ARG2, ARG3, ARG4, ARG5, ARG6, ARG7) \
    (NPError)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPP_NewProcInfo, \

```

```

(ARG1), (ARG2), (ARG3),
(ARG4), (ARG5), (ARG6), (ARG7))
#else

typedef NPErrror  (*NPP_NewUPP) (NPMIMEType pluginType, NPP instance, uint16
mode, int16 argc, char* argn[], char* argv[], NPSavedData* saved);
#define NewNPP_NewProc(FUNC) \
    ((NPP_NewUPP) (FUNC))
#define CallNPP_NewProc(FUNC, ARG1, ARG2, ARG3, ARG4, ARG5, ARG6, ARG7) \
    (*(FUNC))((ARG1), (ARG2), (ARG3), (ARG4), (ARG5), (ARG6), (ARG7))

#endif

/* NPP_Destroy */

#if GENERATINGCFM

typedef UniversalProcPtr NPP_DestroyUPP;
enum {
    uppNPP_DestroyProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPSavedData **)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPErrror)))
};
#define NewNPP_DestroyProc(FUNC) \
    (NPP_DestroyUPP) NewRoutineDescriptor((ProcPtr) (FUNC),
uppNPP_DestroyProcInfo, GetCurrentArchitecture())
#define CallNPP_DestroyProc(FUNC, ARG1, ARG2) \
    (NPErrror) CallUniversalProc((UniversalProcPtr) (FUNC),
uppNPP_DestroyProcInfo, (ARG1), (ARG2))
#else

typedef NPErrror  (*NPP_DestroyUPP) (NPP instance, NPSavedData** save);
#define NewNPP_DestroyProc(FUNC) \
    ((NPP_DestroyUPP) (FUNC))
#define CallNPP_DestroyProc(FUNC, ARG1, ARG2) \
    (*(FUNC))((ARG1), (ARG2))

#endif

/* NPP_SetWindow */

#if GENERATINGCFM

typedef UniversalProcPtr NPP_SetWindowUPP;
enum {
    uppNPP_SetWindowProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPWindow *)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPErrror)))
};
#define NewNPP_SetWindowProc(FUNC) \
    (NPP_SetWindowUPP) NewRoutineDescriptor((ProcPtr) (FUNC),
uppNPP_SetWindowProcInfo, GetCurrentArchitecture())
#define CallNPP_SetWindowProc(FUNC, ARG1, ARG2) \

```

```

        (NPErrror)CallUniversalProc((UniversalProcPtr) (FUNC),
uppNPP_SetWindowProcInfo, (ARG1), (ARG2))

#else

typedef NPErrror    (*NPP_SetWindowUPP) (NPP instance, NPWindow* window);
#define NewNPP_SetWindowProc(FUNC)          \
        ((NPP_SetWindowUPP) (FUNC))
#define CallNPP_SetWindowProc(FUNC, ARG1, ARG2)          \
        (*(FUNC))((ARG1), (ARG2))

#endif

/* NPP_NewStream */

#if GENERATINGCFM

typedef UniversalProcPtr NPP_NewStreamUPP;
enum {
    uppNPP_NewStreamProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPMIMEType)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(NPStream *)))
        | STACK_ROUTINE_PARAMETER(4, SIZE_CODE(sizeof(NPBool)))
        | STACK_ROUTINE_PARAMETER(5, SIZE_CODE(sizeof(uint16 *)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPErrror)))
};
#define NewNPP_NewStreamProc(FUNC)          \
        (NPP_NewStreamUPP) NewRoutineDescriptor((ProcPtr) (FUNC),
uppNPP_NewStreamProcInfo, GetCurrentArchitecture())
#define CallNPP_NewStreamProc(FUNC, ARG1, ARG2, ARG3, ARG4, ARG5)          \
        (NPErrror)CallUniversalProc((UniversalProcPtr) (FUNC),
uppNPP_NewStreamProcInfo, (ARG1), (ARG2), (ARG3), (ARG4), (ARG5))
#else

typedef NPErrror    (*NPP_NewStreamUPP) (NPP instance, NPMIMEType type, NPStream*
stream, NPBool seekable, uint16* stype);
#define NewNPP_NewStreamProc(FUNC)          \
        ((NPP_NewStreamUPP) (FUNC))
#define CallNPP_NewStreamProc(FUNC, ARG1, ARG2, ARG3, ARG4, ARG5) \
        (*(FUNC))((ARG1), (ARG2), (ARG3), (ARG4), (ARG5))

#endif

/* NPP_DestroyStream */

#if GENERATINGCFM

typedef UniversalProcPtr NPP_DestroyStreamUPP;
enum {
    uppNPP_DestroyStreamProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPStream *)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(NPReason)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPErrror)))
};

```

```

#define NewNPP_DestroyStreamProc(FUNC) \
    (NPP_DestroyStreamUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPP_DestroyStreamProcInfo, GetCurrentArchitecture())
#define CallNPP_DestroyStreamProc(FUNC, NPParg, NPStreamPtr, NPReasonArg)
    \
    (NPErrror)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPP_DestroyStreamProcInfo, (NPParg), (NPStreamPtr), (NPReasonArg))

#else

typedef NPErrror    (*NPP_DestroyStreamUPP)(NPP instance, NPStream* stream,
NPReason reason);
#define NewNPP_DestroyStreamProc(FUNC) \
    ((NPP_DestroyStreamUPP)(FUNC))
#define CallNPP_DestroyStreamProc(FUNC, NPParg, NPStreamPtr, NPReasonArg)
    \
    (*(FUNC))((NPParg), (NPStreamPtr), (NPReasonArg))

#endif

/* NPP_WriteReady */

#if GENERATINGCFM

typedef UniversalProcPtr NPP_WriteReadyUPP;
enum {
    uppNPP_WriteReadyProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPStream *)))
        | RESULT_SIZE(SIZE_CODE(sizeof(int32)))
};
#define NewNPP_WriteReadyProc(FUNC) \
    (NPP_WriteReadyUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPP_WriteReadyProcInfo, GetCurrentArchitecture())
#define CallNPP_WriteReadyProc(FUNC, NPParg, NPStreamPtr) \
    (int32)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPP_WriteReadyProcInfo, (NPParg), (NPStreamPtr))

#else

typedef int32 (*NPP_WriteReadyUPP)(NPP instance, NPStream* stream);
#define NewNPP_WriteReadyProc(FUNC) \
    ((NPP_WriteReadyUPP)(FUNC))
#define CallNPP_WriteReadyProc(FUNC, NPParg, NPStreamPtr) \
    (*(FUNC))((NPParg), (NPStreamPtr))

#endif

/* NPP_Write */

#if GENERATINGCFM

typedef UniversalProcPtr NPP_WriteUPP;
enum {
    uppNPP_WriteProcInfo = kThinkCStackBased

```

```

        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPStream *)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(int32)))
        | STACK_ROUTINE_PARAMETER(4, SIZE_CODE(sizeof(int32)))
        | STACK_ROUTINE_PARAMETER(5, SIZE_CODE(sizeof(void*)))
        | RESULT_SIZE(SIZE_CODE(sizeof(int32)))
};
#define NewNPP_WriteProc(FUNC) \
        (NPP_WriteUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPP_WriteProcInfo, GetCurrentArchitecture())
#define CallNPP_WriteProc(FUNC, NPParg, NPStreamPtr, offsetArg, lenArg,
bufferPtr) \
        (int32)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPP_WriteProcInfo, (NPParg), (NPStreamPtr), (offsetArg), (lenArg),
(bufferPtr))

#else

typedef int32 (*NPP_WriteUPP)(NPP instance, NPStream* stream, int32 offset,
int32 len, void* buffer);
#define NewNPP_WriteProc(FUNC) \
        ((NPP_WriteUPP)(FUNC))
#define CallNPP_WriteProc(FUNC, NPParg, NPStreamPtr, offsetArg, lenArg,
bufferPtr) \
        (*(FUNC))((NPParg), (NPStreamPtr), (offsetArg), (lenArg),
(bufferPtr))

#endif

/* NPP_StreamAsFile */

#if GENERATINGCFM

typedef UniversalProcPtr NPP_StreamAsFileUPP;
enum {
        uppNPP_StreamAsFileProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPStream *)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(const char *)))
        | RESULT_SIZE(SIZE_CODE(0))
};
#define NewNPP_StreamAsFileProc(FUNC) \
        (NPP_StreamAsFileUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPP_StreamAsFileProcInfo, GetCurrentArchitecture())
#define CallNPP_StreamAsFileProc(FUNC, ARG1, ARG2, ARG3) \
        (void)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPP_StreamAsFileProcInfo, (ARG1), (ARG2), (ARG3))

#else

typedef void (*NPP_StreamAsFileUPP)(NPP instance, NPStream* stream, const char*
fname);
#define NewNPP_StreamAsFileProc(FUNC) \
        ((NPP_StreamAsFileUPP)(FUNC))
#define CallNPP_StreamAsFileProc(FUNC, ARG1, ARG2, ARG3) \
        (*(FUNC))((ARG1), (ARG2), (ARG3))

```

```

#endif

/* NPP_Print */

#if GENERATINGCFM

typedef UniversalProcPtr NPP_PrintUPP;
enum {
    uppNPP_PrintProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPPrint *)))
        | RESULT_SIZE(SIZE_CODE(0))
};
#define NewNPP_PrintProc(FUNC) \
    (NPP_PrintUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPP_PrintProcInfo, GetCurrentArchitecture())
#define CallNPP_PrintProc(FUNC, NPParg, voidPtr) \
    (void)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPP_PrintProcInfo, (NPParg), (voidPtr))

#else

typedef void (*NPP_PrintUPP)(NPP instance, NPPrint* platformPrint);
#define NewNPP_PrintProc(FUNC) \
    ((NPP_PrintUPP)(FUNC))
#define CallNPP_PrintProc(FUNC, NPParg, NPPrintArg) \
    (*(FUNC))((NPParg), (NPPrintArg))

#endif

/* NPP_HandleEvent */

#if GENERATINGCFM

typedef UniversalProcPtr NPP_HandleEventUPP;
enum {
    uppNPP_HandleEventProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(void *)))
        | RESULT_SIZE(SIZE_CODE(sizeof(int16)))
};
#define NewNPP_HandleEventProc(FUNC) \
    (NPP_HandleEventUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPP_HandleEventProcInfo, GetCurrentArchitecture())
#define CallNPP_HandleEventProc(FUNC, NPParg, voidPtr) \
    (int16)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPP_HandleEventProcInfo, (NPParg), (voidPtr))

#else

typedef int16 (*NPP_HandleEventUPP)(NPP instance, void* event);
#define NewNPP_HandleEventProc(FUNC) \
    ((NPP_HandleEventUPP)(FUNC))
#define CallNPP_HandleEventProc(FUNC, NPParg, voidPtr) \
    (*(FUNC))((NPParg), (voidPtr))

#endif

```



```

#endif

/* NPP_URLNotify */

#if GENERATINGCFM

typedef UniversalProcPtr NPP_URLNotifyUPP;
enum {
    uppNPP_URLNotifyProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(const char*)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(NPReason)))
        | STACK_ROUTINE_PARAMETER(4, SIZE_CODE(sizeof(void*)))
        | RESULT_SIZE(SIZE_CODE(SIZE_CODE(0)))
};
#define NewNPP_URLNotifyProc(FUNC) \
    ((NPP_URLNotifyUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPP_URLNotifyProcInfo, GetCurrentArchitecture()))
#define CallNPP_URLNotifyProc(FUNC, ARG1, ARG2, ARG3, ARG4) \
    (void)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPP_URLNotifyProcInfo, (ARG1), (ARG2), (ARG3), (ARG4))

#else

typedef void (*NPP_URLNotifyUPP)(NPP instance, const char* url, NPReason reason,
void* notifyData);
#define NewNPP_URLNotifyProc(FUNC) \
    ((NPP_URLNotifyUPP)(FUNC))
#define CallNPP_URLNotifyProc(FUNC, ARG1, ARG2, ARG3, ARG4) \
    (*(FUNC))((ARG1), (ARG2), (ARG3), (ARG4))

#endif

/*
 * Netscape entry points
 */

#ifdef XP_UNIX

/* NPN_GetValue */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_GetValueUPP;
enum {
    uppNPN_GetValueProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPNVariable)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(void *)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPError)))
};
#define NewNPN_GetValueProc(FUNC) \

```

```

        (NPN_GetValueUPP) NewRoutineDescriptor((ProcPtr) (FUNC),
uppNPN_GetValueProcInfo, GetCurrentArchitecture())
#define CallNPN_GetURNotifyLProc(FUNC, ARG1, ARG2, ARG3) \
        (NPErrror)CallUniversalProc((UniversalProcPtr) (FUNC),
uppNPN_GetValueProcInfo, (ARG1), (ARG2), (ARG3))
#else

typedef NPErrror    (*NPN_GetValueUPP) (NPP instance, NPNVariable variable, void
*ret_alue);
#define NewNPN_GetValueProc(FUNC) \
        ((NPN_GetValueUPP) (FUNC))
#define CallNPN_GetValueProc(FUNC, ARG1, ARG2, ARG3) \
        (*(FUNC))((ARG1), (ARG2), (ARG3))
#endif

#endif /* XP_UNIX */

/* NPN_GetUrlNotify */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_GetURLNotifyUPP;
enum {
    uppNPN_GetURLNotifyProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(const char*)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(const char*)))
        | STACK_ROUTINE_PARAMETER(4, SIZE_CODE(sizeof(void*)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPErrror)))
};
#define NewNPN_GetURLNotifyProc(FUNC) \
        (NPN_GetURLNotifyUPP) NewRoutineDescriptor((ProcPtr) (FUNC),
uppNPN_GetURLNotifyProcInfo, GetCurrentArchitecture())
#define CallNPN_GetURNotifyLProc(FUNC, ARG1, ARG2, ARG3, ARG4) \
        (NPErrror)CallUniversalProc((UniversalProcPtr) (FUNC),
uppNPN_GetURLNotifyProcInfo, (ARG1), (ARG2), (ARG3), (ARG4))
#else

typedef NPErrror    (*NPN_GetURLNotifyUPP) (NPP instance, const char* url, const
char* window, void* notifyData);
#define NewNPN_GetURLNotifyProc(FUNC) \
        ((NPN_GetURLNotifyUPP) (FUNC))
#define CallNPN_GetURLNotifyProc(FUNC, ARG1, ARG2, ARG3, ARG4) \
        (*(FUNC))((ARG1), (ARG2), (ARG3), (ARG4))
#endif

/* NPN_PostUrlNotify */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_PostURLNotifyUPP;
enum {
    uppNPN_PostURLNotifyProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))

```

```

        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(const char*)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(const char*)))
        | STACK_ROUTINE_PARAMETER(4, SIZE_CODE(sizeof(uint32)))
        | STACK_ROUTINE_PARAMETER(5, SIZE_CODE(sizeof(const char*)))
        | STACK_ROUTINE_PARAMETER(6, SIZE_CODE(sizeof(NPBool)))
        | STACK_ROUTINE_PARAMETER(7, SIZE_CODE(sizeof(void*)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPError)))
};
#define NewNPN_PostURLNotifyProc(FUNC) \
        (NPN_PostURLNotifyUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPN_PostURLNotifyProcInfo, GetCurrentArchitecture())
#define CallNPN_PostURLNotifyProc(FUNC, ARG1, ARG2, ARG3, ARG4, ARG5, ARG6,
ARG7) \
        (NPError)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPN_PostURLNotifyProcInfo, (ARG1), (ARG2), (ARG3), (ARG4), (ARG5), (ARG6),
(ARG7))
#else

typedef NPError (*NPN_PostURLNotifyUPP)(NPP instance, const char* url, const
char* window, uint32 len, const char* buf, NPBool file, void* notifyData);
#define NewNPN_PostURLNotifyProc(FUNC) \
        ((NPN_PostURLNotifyUPP)(FUNC))
#define CallNPN_PostURLNotifyProc(FUNC, ARG1, ARG2, ARG3, ARG4, ARG5, ARG6,
ARG7) \
        (*(FUNC))((ARG1), (ARG2), (ARG3), (ARG4), (ARG5), (ARG6), (ARG7))
#endif

/* NPN_GetUrl */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_GetURLUPP;
enum {
    uppNPN_GetURLProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(const char*)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(const char*)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPError)))
};
#define NewNPN_GetURLProc(FUNC) \
        (NPN_GetURLUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPN_GetURLProcInfo, GetCurrentArchitecture())
#define CallNPN_GetURLProc(FUNC, ARG1, ARG2, ARG3) \
        (NPError)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPN_GetURLProcInfo, (ARG1), (ARG2), (ARG3))
#else

typedef NPError (*NPN_GetURLUPP)(NPP instance, const char* url, const char*
window);
#define NewNPN_GetURLProc(FUNC) \
        ((NPN_GetURLUPP)(FUNC))
#define CallNPN_GetURLProc(FUNC, ARG1, ARG2, ARG3) \
        (*(FUNC))((ARG1), (ARG2), (ARG3))
#endif

```

```

/* NPN_PostUrl */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_PostURLUPP;
enum {
    uppNPN_PostURLProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(const char*)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(const char*)))
        | STACK_ROUTINE_PARAMETER(4, SIZE_CODE(sizeof(uint32)))
        | STACK_ROUTINE_PARAMETER(5, SIZE_CODE(sizeof(const char*)))
        | STACK_ROUTINE_PARAMETER(6, SIZE_CODE(sizeof(NPBool)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPError)))
};
#define NewNPN_PostURLProc(FUNC) \
    (NPN_PostURLUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPN_PostURLProcInfo, GetCurrentArchitecture())
#define CallNPN_PostURLProc(FUNC, ARG1, ARG2, ARG3, ARG4, ARG5, ARG6) \
    (NPError)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPN_PostURLProcInfo, (ARG1), (ARG2), (ARG3), (ARG4), (ARG5), (ARG6))
#else

typedef NPError (*NPN_PostURLUPP)(NPP instance, const char* url, const char*
window, uint32 len, const char* buf, NPBool file);
#define NewNPN_PostURLProc(FUNC) \
    ((NPN_PostURLUPP)(FUNC))
#define CallNPN_PostURLProc(FUNC, ARG1, ARG2, ARG3, ARG4, ARG5, ARG6) \
    (*(FUNC))((ARG1), (ARG2), (ARG3), (ARG4), (ARG5), (ARG6))
#endif

/* NPN_RequestRead */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_RequestReadUPP;
enum {
    uppNPN_RequestReadProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPStream*)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPByteRange*)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPError)))
};
#define NewNPN_RequestReadProc(FUNC) \
    (NPN_RequestReadUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPN_RequestReadProcInfo, GetCurrentArchitecture())
#define CallNPN_RequestReadProc(FUNC, stream, range) \
    (NPError)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPN_RequestReadProcInfo, (stream), (range))
#else

typedef NPError (*NPN_RequestReadUPP)(NPStream* stream, NPByteRange*
rangeList);
#define NewNPN_RequestReadProc(FUNC) \
    ((NPN_RequestReadUPP)(FUNC))
#define CallNPN_RequestReadProc(FUNC, stream, range) \

```

```

        (*(FUNC))((stream), (range))

#endif

/* NPN_NewStream */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_NewStreamUPP;
enum {
    uppNPN_NewStreamProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPMIMETYPE)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(const char *)))
        | STACK_ROUTINE_PARAMETER(4, SIZE_CODE(sizeof(NPStream **)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPError)))
};
#define NewNPN_NewStreamProc(FUNC) \
    (NPN_NewStreamUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPN_NewStreamProcInfo, GetCurrentArchitecture())
#define CallNPN_NewStreamProc(FUNC, npp, type, window, stream) \
    (NPError)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPN_NewStreamProcInfo, (npp), (type), (window), (stream))

#else

typedef NPError    (*NPN_NewStreamUPP)(NPP instance, NPMIMETYPE type, const char*
window, NPStream** stream);
#define NewNPN_NewStreamProc(FUNC) \
    ((NPN_NewStreamUPP)(FUNC))
#define CallNPN_NewStreamProc(FUNC, npp, type, window, stream) \
    (*(FUNC))((npp), (type), (window), (stream))

#endif

/* NPN_Write */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_WriteUPP;
enum {
    uppNPN_WriteProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPStream *)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(int32)))
        | STACK_ROUTINE_PARAMETER(4, SIZE_CODE(sizeof(void*)))
        | RESULT_SIZE(SIZE_CODE(sizeof(int32)))
};
#define NewNPN_WriteProc(FUNC) \
    (NPN_WriteUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPN_WriteProcInfo, GetCurrentArchitecture())
#define CallNPN_WriteProc(FUNC, npp, stream, len, buffer) \
    (int32)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPN_WriteProcInfo, (npp), (stream), (len), (buffer))

```

```

#else

typedef int32 (*NPN_WriteUPP)(NPP instance, NPStream* stream, int32 len, void*
buffer);
#define NewNPN_WriteProc(FUNC) \
    ((NPN_WriteUPP) (FUNC))
#define CallNPN_WriteProc(FUNC, npp, stream, len, buffer) \
    (*(FUNC))((npp), (stream), (len), (buffer))

#endif

/* NPN_DestroyStream */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_DestroyStreamUPP;
enum {
    uppNPN_DestroyStreamProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP )))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPStream *)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(NPReason)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPError)))
};
#define NewNPN_DestroyStreamProc(FUNC) \
    (NPN_DestroyStreamUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPN_DestroyStreamProcInfo, GetCurrentArchitecture())
#define CallNPN_DestroyStreamProc(FUNC, npp, stream, reason) \
    (NPError)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPN_DestroyStreamProcInfo, (npp), (stream), (reason))

#else

typedef NPError (*NPN_DestroyStreamUPP)(NPP instance, NPStream* stream, NPReason
reason);
#define NewNPN_DestroyStreamProc(FUNC) \
    ((NPN_DestroyStreamUPP) (FUNC))
#define CallNPN_DestroyStreamProc(FUNC, npp, stream, reason) \
    (*(FUNC))((npp), (stream), (reason))

#endif

/* NPN_Status */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_StatusUPP;
enum {
    uppNPN_StatusProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(char *)))
};

#define NewNPN_StatusProc(FUNC) \
    (NPN_StatusUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPN_StatusProcInfo, GetCurrentArchitecture())

```

```

#define CallNPN_StatusProc(FUNC, npp, msg) \
    (void)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPN_StatusProcInfo, (npp), (msg))

#else

typedef void (*NPN_StatusUPP)(NPP instance, const char* message);
#define NewNPN_StatusProc(FUNC) \
    ((NPN_StatusUPP)(FUNC))
#define CallNPN_StatusProc(FUNC, npp, msg) \
    (*(FUNC))((npp), (msg))

#endif

/* NPN_UserAgent */
#if GENERATINGCFM

typedef UniversalProcPtr NPN_UserAgentUPP;
enum {
    uppNPN_UserAgentProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | RESULT_SIZE(SIZE_CODE(sizeof(const char *)))
};

#define NewNPN_UserAgentProc(FUNC) \
    (NPN_UserAgentUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPN_UserAgentProcInfo, GetCurrentArchitecture())
#define CallNPN_UserAgentProc(FUNC, ARG1) \
    (const char*)CallUniversalProc((UniversalProcPtr)(FUNC), \
    uppNPN_UserAgentProcInfo, (ARG1))

#else

typedef const char* (*NPN_UserAgentUPP)(NPP instance);
#define NewNPN_UserAgentProc(FUNC) \
    ((NPN_UserAgentUPP)(FUNC))
#define CallNPN_UserAgentProc(FUNC, ARG1) \
    (*(FUNC))((ARG1))

#endif

/* NPN_MemAlloc */
#if GENERATINGCFM

typedef UniversalProcPtr NPN_MemAllocUPP;
enum {
    uppNPN_MemAllocProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(uint32)))
        | RESULT_SIZE(SIZE_CODE(sizeof(void *)))
};

#define NewNPN_MemAllocProc(FUNC) \
    (NPN_MemAllocUPP) NewRoutineDescriptor((ProcPtr)(FUNC), \
    uppNPN_MemAllocProcInfo, GetCurrentArchitecture())
#define CallNPN_MemAllocProc(FUNC, ARG1) \

```

```

        (void*)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPN_MemAllocProcInfo, (ARG1))

#else

typedef void* (*NPN_MemAllocUPP)(uint32 size);
#define NewNPN_MemAllocProc(FUNC) \
        ((NPN_MemAllocUPP)(FUNC))
#define CallNPN_MemAllocProc(FUNC, ARG1) \
        (*(FUNC))((ARG1))

#endif

/* NPN__MemFree */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_MemFreeUPP;
enum {
    uppNPN_MemFreeProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(void *)))
};

#define NewNPN_MemFreeProc(FUNC) \
        (NPN_MemFreeUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPN_MemFreeProcInfo, GetCurrentArchitecture())
#define CallNPN_MemFreeProc(FUNC, ARG1) \
        (void)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPN_MemFreeProcInfo, (ARG1))

#else

typedef void (*NPN_MemFreeUPP)(void* ptr);
#define NewNPN_MemFreeProc(FUNC) \
        ((NPN_MemFreeUPP)(FUNC))
#define CallNPN_MemFreeProc(FUNC, ARG1) \
        (*(FUNC))((ARG1))

#endif

/* NPN_MemFlush */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_MemFlushUPP;
enum {
    uppNPN_MemFlushProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(uint32)))
        | RESULT_SIZE(SIZE_CODE(sizeof(uint32)))
};

#define NewNPN_MemFlushProc(FUNC) \
        (NPN_MemFlushUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPN_MemFlushProcInfo, GetCurrentArchitecture())
#define CallNPN_MemFlushProc(FUNC, ARG1) \

```



```

        (uint32)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPN_MemFlushProcInfo, (ARG1))

#else

typedef uint32 (*NPN_MemFlushUPP)(uint32 size);
#define NewNPN_MemFlushProc(FUNC) \
        ((NPN_MemFlushUPP)(FUNC))
#define CallNPN_MemFlushProc(FUNC, ARG1) \
        (*(FUNC))((ARG1))

#endif

/* NPN_ReloadPlugins */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_ReloadPluginsUPP;
enum {
    uppNPN_ReloadPluginsProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPBool)))
        | RESULT_SIZE(SIZE_CODE(0))
};

#define NewNPN_ReloadPluginsProc(FUNC) \
        (NPN_ReloadPluginsUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPN_ReloadPluginsProcInfo, GetCurrentArchitecture())
#define CallNPN_ReloadPluginsProc(FUNC, ARG1) \
        (void)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPN_ReloadPluginsProcInfo, (ARG1))

#else

typedef void (*NPN_ReloadPluginsUPP)(NPBool reloadPages);
#define NewNPN_ReloadPluginsProc(FUNC) \
        ((NPN_ReloadPluginsUPP)(FUNC))
#define CallNPN_ReloadPluginsProc(FUNC, ARG1) \
        (*(FUNC))((ARG1))

#endif

/* NPN_GetJavaEnv */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_GetJavaEnvUPP;
enum {
    uppNPN_GetJavaEnvProcInfo = kThinkCStackBased
        | RESULT_SIZE(SIZE_CODE(sizeof(JRIEnv*)))
};

#define NewNPN_GetJavaEnvProc(FUNC) \
        (NPN_GetJavaEnvUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPN_GetJavaEnvProcInfo, GetCurrentArchitecture())

```

```

#define CallNPN_GetJavaEnvProc(FUNC) \
    (void)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPN_GetJavaEnvProcInfo)

#else

typedef JRIEnv* (*NPN_GetJavaEnvUPP)(void);
#define NewNPN_GetJavaEnvProc(FUNC) \
    ((NPN_GetJavaEnvUPP)(FUNC))
#define CallNPN_GetJavaEnvProc(FUNC) \
    (*(FUNC))()

#endif

/* NPN_GetJavaPeer */

#if GENERATINGCFM

typedef UniversalProcPtr NPN_GetJavaPeerUPP;
enum {
    uppNPN_GetJavaPeerProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPP)))
        | RESULT_SIZE(SIZE_CODE(sizeof(jref)))
};

#define NewNPN_GetJavaPeerProc(FUNC) \
    (NPN_GetJavaPeerUPP) NewRoutineDescriptor((ProcPtr)(FUNC),
uppNPN_GetJavaPeerProcInfo, GetCurrentArchitecture())
#define CallNPN_GetJavaPeerProc(FUNC, ARG1) \
    (void)CallUniversalProc((UniversalProcPtr)(FUNC),
uppNPN_GetJavaPeerProcInfo, (ARG1))

#else

typedef jref (*NPN_GetJavaPeerUPP)(NPP instance);
#define NewNPN_GetJavaPeerProc(FUNC) \
    ((NPN_GetJavaPeerUPP)(FUNC))
#define CallNPN_GetJavaPeerProc(FUNC, ARG1) \
    (*(FUNC))((ARG1))

#endif

/*****
*****
* The actual plugin function table definitions
*****
*****/

typedef struct _NPPluginFuncs {
    uint16 size;
    uint16 version;
    NPP_NewUPP newp;

```

```

    NPP_DestroyUPP destroy;
    NPP_SetWindowUPP setwindow;
    NPP_NewStreamUPP newstream;
    NPP_DestroyStreamUPP destroystream;
    NPP_StreamAsFileUPP asfile;
    NPP_WriteReadyUPP writeready;
    NPP_WriteUPP write;
    NPP_PrintUPP print;
    NPP_HandleEventUPP event;
    NPP_URLNotifyUPP urlnotify;
    JRIGlobalRef javaClass;
} NPPluginFuncs;

```

```

typedef struct _NPNetscapeFuncs {
    uint16 size;
    uint16 version;
    NPN_GetURLUPP geturl;
    NPN_PostURLUPP posturl;
    NPN_RequestReadUPP requestread;
    NPN_NewStreamUPP newstream;
    NPN_WriteUPP write;
    NPN_DestroyStreamUPP destroystream;
    NPN_StatusUPP status;
    NPN_UserAgentUPP uagent;
    NPN_MemAllocUPP memalloc;
    NPN_MemFreeUPP memfree;
    NPN_MemFlushUPP memflush;
    NPN_ReloadPluginsUPP reloadplugins;
    NPN_GetJavaEnvUPP getJavaEnv;
    NPN_GetJavaPeerUPP getJavaPeer;
    NPN_GetURLNotifyUPP geturlnotify;
    NPN_PostURLNotifyUPP posturlnotify;
#ifdef XP_UNIX
    NPN_GetValueUPP getvalue;
#endif /* XP_UNIX */
} NPNetscapeFuncs;

```

```

#ifdef XP_MAC
/*****
*****
* Mac platform-specific plugin glue stuff
*****
*****/

/*
* Main entry point of the plugin.
* This routine will be called when the plugin is loaded. The function
* tables are passed in and the plugin fills in the NPPluginFuncs table
* and NPPShutdownUPP for Netscape's use.
*/

```

```

#if GENERATINGCFM

```

```

typedef UniversalProcPtr NPP_MainEntryUPP;

```

```

enum {
    uppNPP_MainEntryProcInfo = kThinkCStackBased
        | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(NPNetscapeFuncs*)))
        | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(NPPluginFuncs*)))
        | STACK_ROUTINE_PARAMETER(3, SIZE_CODE(sizeof(NPP_ShutdownUPP*)))
        | RESULT_SIZE(SIZE_CODE(sizeof(NPError)));
};
#define NewNPP_MainEntryProc(FUNC) \
    ((NPP_MainEntryUPP) NewRoutineDescriptor((ProcPtr) (FUNC),
    uppNPP_MainEntryProcInfo, GetCurrentArchitecture()))
#define CallNPP_MainEntryProc(FUNC, netscapeFunc, pluginFunc, shutdownUPP) \
    CallUniversalProc((UniversalProcPtr) (FUNC),
    (ProcInfoType)uppNPP_MainEntryProcInfo, (netscapeFunc), (pluginFunc),
    (shutdownUPP))

#else

typedef NPError (*NPP_MainEntryUPP) (NPNetscapeFuncs*, NPPluginFuncs*,
NPP_ShutdownUPP*);
#define NewNPP_MainEntryProc(FUNC) \
    ((NPP_MainEntryUPP) (FUNC))
#define CallNPP_MainEntryProc(FUNC, netscapeFunc, pluginFunc, shutdownUPP) \
    (*(FUNC))((netscapeFunc), (pluginFunc), (shutdownUPP))

#endif
#endif /* MAC */

#ifdef _WINDOWS

#ifdef __cplusplus
extern "C" {
#endif

/* plugin meta member functions */

NPError WINAPI NP_GetEntryPoints(NPPluginFuncs* pFuncs);

NPError WINAPI NP_Initialize(NPNetscapeFuncs* pFuncs);

NPError WINAPI NP_Shutdown();

#ifdef __cplusplus
}
#endif

#endif /* _WINDOWS */

#ifdef XP_UNIX

#ifdef __cplusplus
extern "C" {
#endif

/* plugin meta member functions */

```

```
char* NP_GetMIMEDescription(void);
NPError NP_Initialize(NPNetscapeFuncs*, NPPluginFuncs*);
NPError NP_Shutdown(void);

#ifdef __cplusplus
}
#endif

#endif /* XP_UNIX */

#endif /* _NPUPP_H_ */
```

```

#include <stdio.h>
// #if defined (_WIN32)^M
// #else
// #include <dlfcn.h>
// #endif
#include "npupp.h"
#include <tcl.h>

extern Tcl_HashTable *dlopen_hashtablePtr;

initPlugin(void *handle) {
    NPNetScapeFuncs netscapeFuncs;
    char f[8];
    NPPluginFuncs pluginFuncs;
    int err;

    pluginFuncs.size = sizeof(NPPluginFuncs);
    pluginFuncs.version = 9;
    pluginFuncs.newp = 0;

    printf("Sizeof Plugin: %d netscapeFuncs: %d\n", sizeof(pluginFuncs),
        sizeof(netscapeFuncs));
    netscapeFuncs.size = sizeof(NPNetScapeFuncs);
    netscapeFuncs.version = 9;
    netscapeFuncs.geturl = 0;

    printf("set sizes\n"); fflush(stdout);

    printf("start init\n"); fflush(stdout);
    err = NP_Initialize(&netscapeFuncs, &pluginFuncs);
    printf("doneinit: %d\n", err); fflush(stdout);
    // NP_GetEntryPoints(&pluginFuncs);

    printf("\nHANDLE: %xx size: %d version: %d newp: %d\n", handle,
        pluginFuncs.size, pluginFuncs.version, pluginFuncs.newp);
    printf("\nHANDLE: %xx size: %d version: %d geturl: %d\n", handle,
        netscapeFuncs.size, netscapeFuncs.version, netscapeFuncs.geturl);
}

```

```

/*
 * plugext1.0 is a compiled version of plugext for Tcl.
 * -- Clif Flynt <clif@cflynt.com> Jan 3, 1999
 * --
 * -- Modified from get.c by
 * -- Jean-Claude Wippler <jcw@equi4.com>, September 17, 1998.
 */

#if defined (_WIN32)
#include <windows.h>
#else
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#endif
#include <dlfcn.h>
#include "npupp.h"
#include "NPN_wrapper.h"
#include "plugextInt.h"

extern Tcl_HashTable *dlopen_hashtablePtr;

Tcl_HashTable *plugext_hashtablePtr;

/* If plugextDebugPrint != 0, then debugprt will print debugging info */
/* This value is set with the default subcommand function debug */

int plugextDebugPrint = 0;

/* This is used to create unique names for un-named plugins */

static int plugextID = 0;

#if defined (_WIN32)
#define WINEXPORT(t) __declspec(dllexport) t
#elif defined (_WIN)
#define WINEXPORT(t) t __export
#else
#define WINEXPORT(t) t
#endif

#if defined (__MWERKS__)
#pragma export on
#endif

FILE *logFl = NULL;

WINEXPORT(int) Plugext_Init(Tcl_Interp* interp)
{
logFl = fopen("LOGFL", "w");

fprintf(logFl, "Hit Plugext_Init\n"); fflush(stdout);
Tcl_CreateObjCommand(interp, "plugext", (Tcl_ObjCmdProc *) plugext_Cmd, 0,
0);
Tcl_PkgProvide(interp, "plugext", "1.0");
}

```

```

    plugext_hashtablePtr = initHashTable();

fprintf(logFl, "Leaving Plugext_Init\n"); fflush(stdout);
    return TCL_OK;
}

WINEXPORT(int) Plugext_SafeInit(Tcl_Interp* interp)
{
    return Plugext_Init(interp);
}

/* ..... */

/*
 * Define the sub commands
 *
 * These strings define the subcommands that the plugext command
 * supports.
 *
 * To add a new subcommand, add the new subcommand string,
 * #define and entry in cmdDefinition
 *
 * Note: Order is important.
 *
 * These are the subcommands that will be recognized
 */

static char *subcommands[] = {
    "create", "debug", "bogus", NULL};

/*
 * These #defines define the positions of the subcommands.
 * You can use enums if you are certain your compiler will provide the
 * same numbers as this.
 */

#define M_create 0
#define M_debug 1
#define M_bogus 2

/*
 * The cmdDefinition structure describes the minimum and maximum number
 * of expected arguments for the subcommand (including cmd and subcommand
 * names), and a usage message to return if the argument
 * count is outside the expected range.
 */

static cmdDefinition definitions[] = {
    {"create ?name? -path path", 3, 7},
    {"debug 0/1", 3, 3},
    {"bogus bogusval", 3, 3}
};

```



```

/* ----- C++ Comment
// These are the segments that need to be declared in the "C" namespace
// These function names and data will be accessed by programs in the
//   Tcl "C" language library.
*/

/* ..... */

/*
-----
*
* plugext_Cmd --
*
*   plugext_Cmd is invoked to process the "plugext" Tcl command.
*   It will parse a subcommand, and perform the requested action.
*
* Results:
*   A standard Tcl result.
*
* Side effects:
*
-----
*/

// WINEXPORT(int)
int plugext_Cmd(ClientData plugext,
                Tcl_Interp *interp,
                int objc,
                Tcl_Obj *objv[]) {

    /* ClientData plugext;          /* Not used. */
    /* Tcl_Interp *interp;          /* Current interpreter. */
    /* int objc;                    /* Number of arguments. */
    /* Tcl_Obj *CONST objv[];       /* Argument objects. */

    int cmdnum;
    int result;
    Tcl_Obj *returnValue;
    CmdReturn *returnStruct;

    /*
    * Initialize the return value
    */

    returnValue = NULL;
    returnStruct = NULL;

    /*
    * Check that we have at least a subcommand,
    * else return an Error and the usage message
    */

    if (objc < 2) {
        Tcl_WrongNumArgs(interp, 1, objv,
            "subcommand ?plugextID? ?fieldName? ?fieldValue?");
        return TCL_ERROR;
    }
}

```

```

/*
 * Find this plugext subcommand in the list of subcommands:
 * Tcl_GetIndexFromObj returns the offset of the recognized string,
 * which is used to index into the command definitions table.
 */

result = Tcl_GetIndexFromObj(interp, objv[1], subcommands,
                             "subcommand", TCL_EXACT, &cmdnum);

/*
 * If the result is not TCL_OK, then the error message is already
 * in the Tcl Interpreter, this code can immediately return.
 */

if (result != TCL_OK) {
    return TCL_ERROR;
}

/*
 * Check that the argument count matches what's expected for this
 * Subcommand.
 */

if ((objc < definitions[cmdnum].minArgCnt) ||
    (objc > definitions[cmdnum].maxArgCnt) ) {
    Tcl_WrongNumArgs(interp, 1, objv, definitions[cmdnum].usage);
    return TCL_ERROR;
}

result = TCL_OK;

/*
 * The subcommand is recognized, and has a valid number of arguments
 * Process the command.
 */

switch (cmdnum) {
    case M_debug: {
        char *tmp;
        tmp = Tcl_GetStringFromObj(objv[2], NULL);
        if (TCL_OK != Tcl_GetInt(interp, tmp, &plugextDebugPrint)) {
            return (TCL_ERROR);
        }
        break;
    }
    case M_create: {
        returnStruct =
            plugext_createCmd((ClientData) NULL, interp, objc, objv);
        break;
    }
    default: {
        char error[80];
        sprintf(error, "Bad sub-command %s. Has no entry in switch",
            Tcl_GetStringFromObj(objv[1], NULL));
        returnValue = Tcl_NewStringObj(error, -1);
    }
}

```

```

        result = TCL_ERROR;
    }
}

/*
 * Extract an object to return from returnStruc.
 * returnStruc will be NULL if the processing is done in this
 * function and no other function is called.
 */

if (returnStruc != NULL) {
    returnValue = returnStruc->object;
    result = returnStruc->status;
    Tcl_Free ((char *)returnStruc);
}

/*
 * Set the return value and return the status
 */

if (returnValue != NULL) {
    Tcl_SetObjResult(interp, returnValue);
}

MyPrint("leaving plugext Cmd\n"); fflush(stdout);

return result;
}

/* ..... */

/*-----
 * int plugext_loadInit (Tcl_Interp *interp, char *hashName) --
 *     Initializes the pluginData structure -
 *     assigns values to NPNetScapeFuncs and NPPluginFuncs
 *
 * Arguments:
 *     interp  The tcl interpreter state pointer
 *     hashName  The key for the hashed PluginData object
 *
 * Results:
 *     Function Pointers are up to date.
 *
 * Side Effects:
 *
 *-----*/
int plugext_loadInit (Tcl_Interp *interp, char *hashName) {
    int err;
    NPNetScapeFuncs *netscapeFuncs;
    NPPluginFuncs *pluginFuncs;
    PluginData *pluginData;

    NPError (*init)();          /* The initialize function */
    NPError (*getent)();        /* The NP_GetEntryPoints function */

    pluginData = (PluginData *) getHashData(interp, hashName,
plugext_hashtablePtr);

```

```

if (pluginData == NULL) {
    return TCL_ERROR;
}

// Initialize these. We'll be using them later.
init = getent = NULL;
err = 0;

pluginData->childList = Tcl_NewListObj(0, NULL);

netscapeFuncs = &(pluginData->netscapeFuncs);
pluginFuncs     = &(pluginData->pluginFuncs);

pluginFuncs->size = sizeof(NPPluginFuncs);
pluginFuncs->version = 9;
pluginFuncs->newp = 0;

netscapeFuncs->size = sizeof(NPNetscapeFuncs);
netscapeFuncs->version = 9;
netscapeFuncs->geturl = 0;

/*
 * NP_GetEntryPoints must be called first
 * initialize the netscapeFuncs structure, then the data is moved
 * to a static 'safe' spot in the call to NP_Initialize. This is
 * not obvious from the docs (or just plain wrong.) (It's clear
 * in the Netscape ONE book. Not the online docs or Programming..Plugins)
 */

setPlugextEntryPoints(netscapeFuncs);
MyPrint("\naddr: %xx size: %d version: %d newp: %xx\n", pluginFuncs,
pluginFuncs->size, pluginFuncs->version, pluginFuncs->newp);
MyPrint("\naddr: %xx size: %d version: %d geturl: %xx\n", netscapeFuncs,
netscapeFuncs->size, netscapeFuncs->version, netscapeFuncs->geturl);

#if defined (_WIN32)
    init = (NPError (*)()) GetProcAddress(pluginData->dlopenHandle,
"NP_Initialize");
    if (init == NULL) {
        init = (NPError (*)()) GetProcAddress(pluginData->dlopenHandle,
"NP_PluginInit");
    }
    getent = (NPError (*)()) GetProcAddress(pluginData->dlopenHandle,
"NP_GetEntryPoints");
#else
    init = dlsym(pluginData->dlopenHandle, "NP_Initialize");
    if (init == NULL) {
        init = dlsym(pluginData->dlopenHandle, "NP_PluginUnixInit");
    }
#endif

if (init == NULL) {
    // dprintf("FAILED dlsym: %s\n", dlerror());
    setErrorReturns(interp,
        "Unable to load plugin. Can't find NP_Initialize\n", NULL, NULL, NULL,

```

```

        "Unable to load plugin. Can't find NP_Initialize\n", NULL, NULL, NULL,
        "plugext_loadInit", "510");
        // return TCL_ERROR;
    }

#ifdef _WIN32

    getent = (NPError (*)( )) GetProcAddress(pluginData->dlopenHandle,
    "NP_GetEntryPoints");

    if (getent == NULL) {
        setErrorReturns(interp,
        "Unable to load plugin. Can't find NP_GetEntryPoints\n", NULL,
        NULL, NULL,
        "Unable to load plugin. Can't find NP_GetEntryPoints\n", NULL,
        NULL, NULL,
        "plugext_loadInit", "511");
        // dprintf("FAILED dlsym: %s\n", dlerror());
        return TCL_ERROR;
    }

    if (getent != NULL ) {
        fprintf(logFl, "getent next\n"); fflush(stdout);
        err = getent(pluginFuncs);
        fprintf(logFl, "getent DONE :err %d\n", err); fflush(stdout);
        if (err != NPERR_NO_ERROR) {
            return TCL_ERROR;
        }
    }

    fprintf(logFl, "init next\n"); fflush(stdout);
    if ((init != NULL) && (err == NPERR_NO_ERROR)) {
        err = init(netscapeFuncs);
    }

#else

    if (getent != NULL ) {
        fprintf(logFl, "init DONE - getent next\n"); fflush(stdout);
        err = getent(pluginFuncs);
        fprintf(logFl, "getent DONE \n"); fflush(stdout);
    }

    if ((init != NULL) && (err == 0)) {
        err = init(netscapeFuncs, pluginFuncs);
    }

#endif

    if (err != NPERR_NO_ERROR) {
        return TCL_ERROR;
    }

    fprintf(logFl, "init done\n"); fflush(stdout);

    MyPrint("\n size: %d version: %d newp: %xx\n", pluginFuncs->size,
    pluginFuncs->version, pluginFuncs->newp);

```

```

    MyPrint("\n size: %d version: %d geturl: %xx\n", netscapeFuncs->size,
netscapeFuncs->version, netscapeFuncs->geturl);

```

```

    MyPrint("Blowing this popsicle stand: loadInit\n"); fflush(stdout);

```

```

    return TCL_OK;
}

```

```

/*-----
* CmdReturn *plugext_createCmd ()--
*   Loads a plugin.so file.
#   Initializes the Function pointer tables.
#   Creates a new command to access the new plugin.
* Arguments
*   objv[0]:      "plugext"
*   objv[1]:      "create"
*   objv[2]:      ?pluginID?
*   objv[3]:      "-path"
*   objv[4]:      "path/to/plugin.so"
*
* Results
*   Creates a new hash entry.  Sets error if entry already exists.
*
* Side Effects:
*   None
-----*/

```

```

CmdReturn *plugext_createCmd (ClientData info,
                               Tcl_Interp *interp,
                               int objc,
                               Tcl_Obj *objv[]) {
    Tcl_HashEntry *hashEntryPtr;
    CmdReturn *returnStructPtr;
    char *returnCharPtr = NULL;
    Tcl_Obj *returnObjPtr = NULL;

```

```

//    char *hashKeyPtr;
    int isNew;
    char hashName[80];
    PluginData *pluginData;

```

```

    char *pluginPath;
    char *tmp;
    int i;

```

```

/* If the first arg starts with a "-", its a flag and there is no
 * hardcoded ID name (we choose one).  If not, we grab that name.
 */

```

```

tmp = Tcl_GetStringFromObj(objv[2], NULL);

```

```

if (*tmp != '-') {
    strcpy(hashName, Tcl_GetStringFromObj(objv[2], NULL));
} else {
    sprintf(hashName, "%s%d", "plugext", plugextID++);
}

```

```

/*

```

```

    * Print that the function was called for debugging
    */

    MyPrint("plugext_createCmd called with %d args : hashName: %s\n", objc,
hashName);

    /*
    * create a return structure.
    */

    returnStructPtr = (CmdReturn *) makeCmdReturn();

    /* Initialize these parameters before we check for program settings */

    pluginPath = NULL;

    for (i=0; i<objc; i++) {
        if (!strcmp("-path", Tcl_GetStringFromObj(objv[i], NULL))) {
            char *tmp;
            i++;
            tmp = Tcl_GetStringFromObj(objv[i], NULL);
            pluginPath = Tcl_Alloc (strlen(tmp)+1);
            strcpy(pluginPath, tmp);
        }
    }

    if (pluginPath == NULL) {
        setErrorReturns(interp,
            "pluginName create must include -path", "", "", "",
            "No -path option", "", "", "",
            "pluginName create", "501");

        returnStructPtr->status = TCL_ERROR;
        goto done;
    }

    /*
    * Extract the string representation of the object argument,
    * and use that as a key into the hash table.
    *
    * If this entry already exists, complain.
    */

    hashEntryPtr = Tcl_CreateHashEntry(plugext_hashtablePtr, hashName, &isNew);

    if (!isNew) {
        setErrorReturns(interp,
            "Plugin named \"%s\" already exists.\n", hashName, "", "",
            "plugin with this name exists", "", "", "",
            "plugext create", "500");

        MyPrint("failed on isNew");

        returnStructPtr->status = TCL_ERROR;
        goto done;
    }

```

```

/*
 * If we are here, then the key is unused.
 * Get the string representation from the object,
 * and make a copy that can be placed into the hash table.
 */

fprintf(logFl, "SIZE of PluginData: %d NPN: %d NPP: %d\n",
        sizeof(PluginData), sizeof(NPNetscapeFuncs), sizeof(NPPluginFuncs));

pluginData = (PluginData *) Tcl_Alloc (sizeof(PluginData));
Tcl_SetHashValue(hashEntryPtr, pluginData);

// dprintf("dlopen Next!: %s :: %xx :: %xx \n", pluginPath, NP_Initialize,
NPP_New);

{
char *tmp;
tmp = strrchr(pluginPath, '/');
if (tmp == NULL) {
    tmp = strrchr(pluginPath, '\\');
}
if (tmp == NULL) {
    tmp = pluginPath;
}
pluginData->DLLname = tmp;
}

#if defined (_WIN32)
pluginData->dlopenHandle = LoadLibrary(pluginPath);
fprintf(logFl, "LoadLib: %xx\n", pluginData->dlopenHandle);
fprintf(logFl, "DLLNAME:: %s\n", pluginData->DLLname);
fprintf(stderr, "LoadLib: %xx\n", pluginData->dlopenHandle);

{
HMODULE hModule;
char name[255];
DWORD hInfo;
DWORD dwLen;

struct {
    WORD    first;
    WORD    second;
} *lpBuffer;

hModule = GetModuleHandle(pluginData->DLLname);

if (hModule == NULL) {
    setErrorReturns(interp,
        "Failed to get Module Handle for: %s\n", pluginData->DLLname,
        NULL, NULL,
        "Failed to get Module Handle for: %s\n", pluginData->DLLname,
        NULL, NULL,
        "plugextName_create", "515");
    // returnStructPtr->status = TCL_ERROR;
    pluginData->VersionData = (void *)Tcl_Alloc(8);
}

```



```

        strcpy(pluginData->VersionData, "-1.0");
    } else {

        GetModuleFileName(hModule, name, 255);
        dwLen = GetFileVersionInfoSize(name, &hInfo);

        if (dwLen == 0) {
            setErrorReturns(interp,
                "Failed to get dwLen", NULL, NULL, NULL,
                "Failed to get dwLen", NULL, NULL, NULL,
                "plugextName_getmimeCmd", "515");
            returnStructPtr->status = TCL_ERROR;
            goto done;
        }

        pluginData->VersionData = (void *)Tcl_Alloc(dwLen);

        GetFileVersionInfo(name, hInfo, dwLen, pluginData->VersionData);

        VerQueryValue(pluginData->VersionData, "\\VarFileInfo\\Translation",
            (LPVOID*) &lpBuffer, (unsigned int *) &dwLen);

        if (dwLen == 0) {
            setErrorReturns(interp,
                "Failed to get second dwLen", NULL, NULL, NULL,
                "Failed to get second dwLen", NULL, NULL, NULL,
                "plugextName_getmimeCmd", "516");
            returnStructPtr->status = TCL_ERROR;
            goto done;
        }
    }
}
#else
    pluginData->dlopenHandle = dlopen(pluginPath, RTLD_NOW | RTLD_GLOBAL);
    if (pluginData->dlopenHandle == NULL) {
        setErrorReturns(interp,
            "Unable to load %s.\n %s\n", pluginPath, dlerror(), NULL,
            "Unable to load %s.\n %s\n", pluginPath, dlerror(), NULL,
            "plugext_createCmd", "512");
        returnStructPtr->status = TCL_ERROR;
        goto done;
    }
#endif

// dprintf("dlopen DONE!: %s :: %xx :: %xx \n", pluginPath, NP_Initialize,
NPP_New);
//dprintf("dlopen DONE!: %s :: %xx :: %xx \n", pluginPath, dlsym(pluginData-
>dlopenHandle, "NP_Initialize"), dlsym(pluginData->dlopenHandle, "NPP_New"));

MyPrint("set functPtr: %xx\n", pluginData);

/* Initialize the function pointers */

returnStructPtr->status = plugext_loadInit(interp, hashName);

```

```

    if (returnStructPtr->status != TCL_OK) {goto done;}

done:

    if (returnStructPtr->status == TCL_OK) {
        returnCharPtr = hashName;
        Tcl_CreateObjCommand(interp, hashName,
            (Tcl_ObjCmdProc *) plugextName_Cmd, 0, 0);
    } else {
        returnCharPtr = "Failed to Load extension";
    }

    if ((returnObjPtr == NULL)) {
        returnObjPtr = Tcl_NewStringObj(returnCharPtr, -1);
    }

    returnStructPtr->object = returnObjPtr;

    MyPrint("Leaving plugext createCmd \n"); fflush(stdout);

    return returnStructPtr;
}

/**
NPError      NP_LOADDS      NPP_New(NPMIMEType pluginType, NPP instance,
                                uint16 mode, int16 argc, char* argn[],
                                char* argv[], NPSaveData* saved) {
    fprintf(logFl, "This is a NEW dummy NPP_New to be overwritten\n");
}

NPError NP_Initialize(
    NPNetScapeFuncs *netscapeFuncs,
    NPPluginFuncs *pluginFuncs) {
    fprintf(logFl, "This is a NEW dummy NP_Initialize to be overwritten\n");
}
**/

```

```

# Microsoft Developer Studio Project File - Name="plugext" - Package Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version 5.00
# ** DO NOT EDIT **

# TARGETTYPE "Win32 (x86) Dynamic-Link Library" 0x0102

CFG=plugext - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "plugext.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For example:
!MESSAGE
!MESSAGE NMAKE /f "plugext.mak" CFG="plugext - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "plugext - Win32 Release" (based on\
"Win32 (x86) Dynamic-Link Library")
!MESSAGE "plugext - Win32 Debug" (based on "Win32 (x86) Dynamic-Link Library")
!MESSAGE

# Begin Project
# PROP Scc_ProjName ""
# PROP Scc_LocalPath ""
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF "$ (CFG)" == "plugext - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /YX
/FD /c
# ADD CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /YX /FD
/c
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe

```

```

# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib
/nologo /subsystem:windows /dll /machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib
/nologo /subsystem:windows /dll /machine:I386

!ELSEIF "$(CFG)" == "plugext - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_WINDOWS" /YX /FD /c
# ADD CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /I "C:\tcl805\include" /D "WIN32" /D
"_DEBUG" /D "_WINDOWS" /YX /FD /c
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib
/nologo /subsystem:windows /dll /debug /machine:I386 /pdbtype:sept
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib
C:\tcl805\lib\tcl80.lib C:\tcl805\lib\tk80.lib version.lib /nologo
/subsystem:windows /dll /debug /machine:I386 /pdbtype:sept

!ENDIF

# Begin Target

# Name "plugext - Win32 Release"
# Name "plugext - Win32 Debug"
# Begin Source File

SOURCE=.\extensionLib.c
# End Source File
# Begin Source File

SOURCE=.\extensionLib.h
# End Source File
# Begin Source File

SOURCE=.\NPN_wrapper.c

```

```
# End Source File
# Begin Source File

SOURCE=.\NPN_wrapper.h
# End Source File
# Begin Source File

SOURCE=.\npupp.h
# End Source File
# Begin Source File

SOURCE=.\plugext.c
# End Source File
# Begin Source File

SOURCE=.\plugextCmd.c
# End Source File
# Begin Source File

SOURCE=.\plugextCmdAZ.c
# End Source File
# Begin Source File

SOURCE=.\plugextInt.h
# End Source File
# Begin Source File

SOURCE=.\plugInstCmd.c
# End Source File
# Begin Source File

SOURCE=.\plugInstCmdAZ.c
# End Source File
# Begin Source File

SOURCE=.\tcl805.h
# End Source File
# End Target
# End Project
```

Microsoft Developer Studio Workspace File, Format Version 5.00
WARNING: DO NOT EDIT OR DELETE THIS WORKSPACE FILE!

#####

Project: "plugext"=".\\plugext.dsp - Package Owner=<4>

Package=<5>

{{{
}}}

Package=<4>

{{{
}}}

#####

Global:

Package=<5>

{{{
}}}

Package=<3>

{{{
}}}

#####

```

# Microsoft Developer Studio Generated NMAKE File, Based on plugext.dsp
!IF "$(CFG)" == ""
CFG=plugext - Win32 Debug
!MESSAGE No configuration specified. Defaulting to plugext - Win32 Debug.
!ENDIF

!IF "$(CFG)" != "plugext - Win32 Release" && "$(CFG)" !=\
"plugext - Win32 Debug"
!MESSAGE Invalid configuration "$(CFG)" specified.
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For example:
!MESSAGE
!MESSAGE NMAKE /f "plugext.mak" CFG="plugext - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "plugext - Win32 Release" (based on\
"Win32 (x86) Dynamic-Link Library")
!MESSAGE "plugext - Win32 Debug" (based on "Win32 (x86) Dynamic-Link Library")
!MESSAGE
!ERROR An invalid configuration is specified.
!ENDIF

!IF "$(OS)" == "Windows_NT"
NULL=
!ELSE
NULL=nul
!ENDIF

CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF "$(CFG)" == "plugext - Win32 Release"

OUTDIR=.\Release
INTDIR=.\Release
# Begin Custom Macros
OutDir=.\Release
# End Custom Macros

!IF "$(RECURSE)" == "0"

ALL : "$(OUTDIR)\plugext.dll"

!ELSE

ALL : "$(OUTDIR)\plugext.dll"

!ENDIF

CLEAN :
    -@erase "$(INTDIR)\extensionLib.obj"
    -@erase "$(INTDIR)\NPN_wrapper.obj"
    -@erase "$(INTDIR)\plugext.obj"
    -@erase "$(INTDIR)\plugextCmd.obj"
    -@erase "$(INTDIR)\plugextCmdAZ.obj"

```

```

-@erase "$(INTDIR)\plugInstCmd.obj"
-@erase "$(INTDIR)\plugInstCmdAZ.obj"
-@erase "$(INTDIR)\vc50.idb"
-@erase "$(OUTDIR)\plugext.dll"
-@erase "$(OUTDIR)\plugext.exp"
-@erase "$(OUTDIR)\plugext.lib"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

CPP_PROJ=/nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" \
/Fp"$(INTDIR)\plugext.pch" /YX /Fo"$(INTDIR)\\" /Fd"$(INTDIR)\\" /FD /c
CPP_OBJS=.\Release/
CPP_SBRS=.
MTL_PROJ=/nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
BSC32=bscmake.exe
BSC32_FLAGS=/nologo /o"$(OUTDIR)\plugext.bsc"
BSC32_SBRS= \

LINK32=link.exe
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
odbc32.lib /nologo /subsystem:windows /dll /incremental:no\
/pdb:"$(OUTDIR)\plugext.pdb" /machine:I386 /out:"$(OUTDIR)\plugext.dll"\
/implib:"$(OUTDIR)\plugext.lib"
LINK32_OBJS= \
    "$(INTDIR)\extensionLib.obj" \
    "$(INTDIR)\NPN_wrapper.obj" \
    "$(INTDIR)\plugext.obj" \
    "$(INTDIR)\plugextCmd.obj" \
    "$(INTDIR)\plugextCmdAZ.obj" \
    "$(INTDIR)\plugInstCmd.obj" \
    "$(INTDIR)\plugInstCmdAZ.obj"

"$(OUTDIR)\plugext.dll" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
    $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ELSEIF "$(CFG)" == "plugext - Win32 Debug"

OUTDIR=.\Debug
INTDIR=.\Debug
# Begin Custom Macros
OutDir=.\Debug
# End Custom Macros

!IF "$(RECURSE)" == "0"

ALL : "$(OUTDIR)\plugext.dll"

!ELSE

ALL : "$(OUTDIR)\plugext.dll"

!ENDIF

```



```

CLEAN :
    -@erase "$(INTDIR)\extensionLib.obj"
    -@erase "$(INTDIR)\NPN_wrapper.obj"
    -@erase "$(INTDIR)\plugext.obj"
    -@erase "$(INTDIR)\plugextCmd.obj"
    -@erase "$(INTDIR)\plugextCmdAZ.obj"
    -@erase "$(INTDIR)\plugInstCmd.obj"
    -@erase "$(INTDIR)\plugInstCmdAZ.obj"
    -@erase "$(INTDIR)\vc50.idb"
    -@erase "$(INTDIR)\vc50.pdb"
    -@erase "$(OUTDIR)\plugext.dll"
    -@erase "$(OUTDIR)\plugext.exp"
    -@erase "$(OUTDIR)\plugext.ilc"
    -@erase "$(OUTDIR)\plugext.lib"
    -@erase "$(OUTDIR)\plugext.pdb"

"$ (OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

CPP_PROJ=/nologo /MTd /W3 /Gm /GX /Zi /Od /I "C:\tcl805\include" /D "WIN32" /D\
    "_DEBUG" /D "_WINDOWS" /Fp"$(INTDIR)\plugext.pch" /YX /Fo"$(INTDIR)\\"
    /Fd"$(INTDIR)\\" /FD /c
CPP_OBJS=.\Debug/
CPP_SBRS=.
MTL_PROJ=/nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
BSC32=bscmake.exe
BSC32_FLAGS=/nologo /o"$(OUTDIR)\plugext.bsc"
BSC32_SBRS= \

LINK32=link.exe
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
    advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
    odbccp32.lib C:\tcl805\lib\tcl80.lib C:\tcl805\lib\tk80.lib version.lib\
    /nologo /subsystem:windows /dll /incremental:yes /pdb:"$(OUTDIR)\plugext.pdb"\
    /debug /machine:I386 /out:"$(OUTDIR)\plugext.dll"\
    /implib:"$(OUTDIR)\plugext.lib" /pdbtype:sept
LINK32_OBJS= \
    "$(INTDIR)\extensionLib.obj" \
    "$(INTDIR)\NPN_wrapper.obj" \
    "$(INTDIR)\plugext.obj" \
    "$(INTDIR)\plugextCmd.obj" \
    "$(INTDIR)\plugextCmdAZ.obj" \
    "$(INTDIR)\plugInstCmd.obj" \
    "$(INTDIR)\plugInstCmdAZ.obj"

"$ (OUTDIR)\plugext.dll" : "$ (OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
    $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ENDIF

.c{$ (CPP_OBJS)}.obj::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<

```

```
.cpp{$(CPP_OBJS)}.obj::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
```

```
.cxx{$(CPP_OBJS)}.obj::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
```

```
.c{$(CPP_SBRS)}.sbr::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
```

```
.cpp{$(CPP_SBRS)}.sbr::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
```

```
.cxx{$(CPP_SBRS)}.sbr::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
```

```
!IF "$(CFG)" == "plugext - Win32 Release" || "$(CFG)" ==\
"plugext - Win32 Debug"
SOURCE=.\extensionLib.c
```

```
!IF "$(CFG)" == "plugext - Win32 Release"
```

```
DEP_CPP_EXTEN=\
    ".\extensionLib.h"\
    ".\tcl805.h"
```

```
"$(INTDIR)\extensionLib.obj" : $(SOURCE) $(DEP_CPP_EXTEN) "$(INTDIR)"
```

```
!ELSEIF "$(CFG)" == "plugext - Win32 Debug"
```

```
DEP_CPP_EXTEN=\
    ".\extensionLib.h"\
    ".\tcl805.h"
```

```
"$(INTDIR)\extensionLib.obj" : $(SOURCE) $(DEP_CPP_EXTEN) "$(INTDIR)"
```

```
!ENDIF
```

```
SOURCE=.\NPN_wrapper.c
```

```
!IF "$(CFG)" == "plugext - Win32 Release"
```

```
DEP_CPP_NPN_W=\
```

```

        ".\extensionLib.h"\
        ".\jri.h"\
        ".\jri_md.h"\
        ".\npapi.h"\
        ".\npupp.h"\
        ".\plugextInt.h"\
        ".\tcl805.h"\

"$ (INTDIR)\NPN_wrapper.obj" : $ (SOURCE) $ (DEP_CPP_NPN_W) "$ (INTDIR) "

!ELSEIF "$ (CFG)" == "plugext - Win32 Debug"

DEP_CPP_NPN_W=\
    ".\extensionLib.h"\
    ".\jri.h"\
    ".\jri_md.h"\
    ".\npapi.h"\
    ".\npupp.h"\
    ".\plugextInt.h"\
    ".\tcl805.h"\

"$ (INTDIR)\NPN_wrapper.obj" : $ (SOURCE) $ (DEP_CPP_NPN_W) "$ (INTDIR) "

!ENDIF

SOURCE=.\plugext.c

!IF "$ (CFG)" == "plugext - Win32 Release"

DEP_CPP_PLUGE=\
    ".\extensionLib.h"\
    ".\jri.h"\
    ".\jri_md.h"\
    ".\npapi.h"\
    ".\NPN_wrapper.h"\
    ".\npupp.h"\
    ".\plugextInt.h"\
    ".\tcl805.h"\

"$ (INTDIR)\plugext.obj" : $ (SOURCE) $ (DEP_CPP_PLUGE) "$ (INTDIR) "

!ELSEIF "$ (CFG)" == "plugext - Win32 Debug"

DEP_CPP_PLUGE=\
    ".\extensionLib.h"\
    ".\jri.h"\
    ".\jri_md.h"\
    ".\npapi.h"\
    ".\NPN_wrapper.h"\
    ".\npupp.h"\
    ".\plugextInt.h"\

```

```

        "..\tcl805.h"\

"$ (INTDIR)\plugext.obj" : $(SOURCE) $(DEP_CPP_PLUGE) "$ (INTDIR) "

!ENDIF

SOURCE=.\plugextCmd.c

!IF "$ (CFG)" == "plugext - Win32 Release"

DEP_CPP_PLUGEX=\
    "..\extensionLib.h"\
    "..\jri.h"\
    "..\jri_md.h"\
    "..\npapi.h"\
    "..\npupp.h"\
    "..\plugextInt.h"\
    "..\tcl805.h"\

"$ (INTDIR)\plugextCmd.obj" : $(SOURCE) $(DEP_CPP_PLUGEX) "$ (INTDIR) "

!ELSEIF "$ (CFG)" == "plugext - Win32 Debug"

DEP_CPP_PLUGEX=\
    "..\extensionLib.h"\
    "..\jri.h"\
    "..\jri_md.h"\
    "..\npapi.h"\
    "..\npupp.h"\
    "..\plugextInt.h"\
    .. "\tcl805.h"\

"$ (INTDIR)\plugextCmd.obj" : $(SOURCE) $(DEP_CPP_PLUGEX) "$ (INTDIR) "

!ENDIF

SOURCE=.\plugextCmdAZ.c

!IF "$ (CFG)" == "plugext - Win32 Release"

DEP_CPP_PLUGEXT=\
    "..\extensionLib.h"\
    "..\jri.h"\
    "..\jri_md.h"\
    "..\npapi.h"\
    "..\npupp.h"\
    "..\plugextInt.h"\
    "..\tcl805.h"\

"$ (INTDIR)\plugextCmdAZ.obj" : $(SOURCE) $(DEP_CPP_PLUGEXT) "$ (INTDIR) "

```

```
!ELSEIF "$(CFG)" == "plugext - Win32 Debug"
```

```
DEP_CPP_PLUGEXT=\
    ".\extensionLib.h"\
    ".\jri.h"\
    ".\jri_md.h"\
    ".\npapi.h"\
    ".\npupp.h"\
    ".\plugextInt.h"\
    ".\tcl805.h"
```

```
"$(INTDIR)\plugextCmdAZ.obj" : $(SOURCE) $(DEP_CPP_PLUGEXT) "$(INTDIR) "
```

```
!ENDIF
```

```
SOURCE=.\plugInstCmd.c
```

```
!IF "$(CFG)" == "plugext - Win32 Release"
```

```
DEP_CPP_PLUGI=\
    ".\extensionLib.h"\
    ".\jri.h"\
    ".\jri_md.h"\
    ".\npapi.h"\
    ".\npupp.h"\
    ".\plugextInt.h"\
    ".\tcl805.h"
```

```
"$(INTDIR)\plugInstCmd.obj" : $(SOURCE) $(DEP_CPP_PLUGI) "$(INTDIR) "
```

```
!ELSEIF "$(CFG)" == "plugext - Win32 Debug"
```

```
DEP_CPP_PLUGI=\
    ".\extensionLib.h"\
    ".\jri.h"\
    ".\jri_md.h"\
    ".\npapi.h"\
    ".\npupp.h"\
    ".\plugextInt.h"\
    ".\tcl805.h"
```

```
"$(INTDIR)\plugInstCmd.obj" : $(SOURCE) $(DEP_CPP_PLUGI) "$(INTDIR) "
```

```
!ENDIF
```

```
SOURCE=.\plugInstCmdAZ.c
```

```
!IF "$(CFG)" == "plugext - Win32 Release"
```

```
DEP_CPP_PLUGIN=\
    ".\extensionLib.h"\
    ".\jri.h"\
    ".\jri_md.h"\
    ".\npapi.h"\
    ".\npupp.h"\
    ".\plugextInt.h"\
    ".\tcl805.h"
```

```
NODEP_CPP_PLUGIN=\
    ".\tk.h"
```

```
"$(INTDIR)\plugInstCmdAZ.obj" : $(SOURCE) $(DEP_CPP_PLUGIN) "$(INTDIR)"
```

```
!ELSEIF "$(CFG)" == "plugext - Win32 Debug"
```

```
DEP_CPP_PLUGIN=\
    ".\extensionLib.h"\
    ".\jri.h"\
    ".\jri_md.h"\
    ".\npapi.h"\
    ".\npupp.h"\
    ".\plugextInt.h"\
    ".\tcl805.h"\
    "c:\tcl805\include\tk.h"\
    "c:\tcl805\include\x11\x.h"\
    "c:\tcl805\include\x11\xfuncproto.h"\
    "c:\tcl805\include\x11\xlib.h"
```

```
"$(INTDIR)\plugInstCmdAZ.obj" : $(SOURCE) $(DEP_CPP_PLUGIN) "$(INTDIR)"
```

```
!ENDIF
```

```
!ENDIF
```

-----Configuration: plugext - Win32 Debug-----
Beginning build with project "F:\Win\Plugext\plugext.dsp", at root.
Active configuration is Win32 (x86) Dynamic-Link Library (based on Win32 (x86) Dynamic-Link Library)

Project's tools are:

"32-bit C/C++ Compiler for 80x86" with flags "/nologo /MTd /W3 /Gm /GX /Zi /Od /I "C:\tcl805\include" /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /Fp"Debug\plugext.pch" /YX /Fo"Debug/" /Fd"Debug/" /FD /c "
"OLE Type Library Maker" with flags "/nologo /D "_DEBUG" /mktyplib203 /o NUL /win32 "
"Win32 Resource Compiler" with flags "/l 0x409 /d "_DEBUG" "
"Browser Database Maker" with flags "/nologo /o"Debug\plugext.bsc" "
"COFF Linker for 80x86" with flags "kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib C:\tcl805\lib\tcl80.lib C:\tcl805\lib\tk80.lib version.lib /nologo /subsystem:windows /dll /incremental:yes /pdb:"Debug\plugext.pdb" /debug /machine:I386 /out:"Debug\plugext.dll" /implib:"Debug\plugext.lib" /pdbtype:sept "
"Custom Build" with flags ""
"<Component 0xa>" with flags ""

Creating temp file "C:\WINDOWS\TEMP\RSPD0D5.TMP" with contents </nologo /MTd /W3 /Gm /GX /Zi /Od /I "C:\tcl805\include" /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /Fp"Debug\plugext.pch" /YX /Fo"Debug/" /Fd"Debug/" /FD /c "F:\Win\Plugext\plugInstCmdAZ.c"

>
Creating command line "cl.exe @C:\WINDOWS\TEMP\RSPD0D5.TMP"
Creating temp file "C:\WINDOWS\TEMP\RSPD0D6.TMP" with contents <kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib C:\tcl805\lib\tcl80.lib C:\tcl805\lib\tk80.lib version.lib /nologo /subsystem:windows /dll /incremental:yes /pdb:"Debug\plugext.pdb" /debug /machine:I386 /out:"Debug\plugext.dll" /implib:"Debug\plugext.lib" /pdbtype:sept .\Debug\extensionLib.obj .\Debug\NPN_wrapper.obj .\Debug\plugext.obj .\Debug\plugextCmd.obj .\Debug\plugextCmdAZ.obj .\Debug\plugInstCmd.obj .\Debug\plugInstCmdAZ.obj>
Creating command line "link.exe @C:\WINDOWS\TEMP\RSPD0D6.TMP"

Compiling...

plugInstCmdAZ.c

F:\Win\Plugext\plugInstCmdAZ.c(89) : warning C4013: 'MyPrint' undefined; assuming extern returning int

F:\Win\Plugext\plugInstCmdAZ.c(826) : warning C4133: 'function' : incompatible types - from 'int *' to 'unsigned short *'

Linking...

plugext.dll - 0 error(s), 2 warning(s)

```

/*
 * plugextCmd.c --
 * Copyright (c) 1997 Clif Flynt
 * All rights reserved.
 *
 * Permission is hereby granted, without written agreement and without
 * license or royalty fees, to use, copy, modify, and distribute this
 * software and its documentation for any purpose, provided that the
 * above copyright notice and the following two paragraphs appear in
 * all copies of this software.
 *
 * IN NO EVENT SHALL Clif Flynt BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * Clif Flynt SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND Clif Flynt HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
 */
/* Include the usual suspects. */

#ifdef _WIN32
#include <windows.h>
#endif
#include "stdlib.h"
#include "ctype.h"
#include "plugextInt.h"

#define plugextCMD_REVISION "$Revision: 1.9 $"

extern int plugextDebugPrint;

/*
 * Define the sub commands
 *
 * These strings define the subcommands that the plugext command
 * supports.
 *
 * To add a new subcommand, add the new subcommand string,
 * #define and entry in cmdDefinition
 *
 * Note: Order is important.
 *
 * These are the subcommands that will be recognized
 */

static char *subcommands[] = {
    "new", "destroy", "children", "getmime", "bogus", NULL};

/*
 * These #defines define the positions of the subcommands.
 * You can use enums if you are certain your compiler will provide the
 * same numbers as this.

```



```

*/

#define M_new 0
#define M_destroy 1
#define M_children 2
#define M_getmime 3
#define M_bogus 4

/*
 * The cmdDefinition structure describes the minimum and maximum number
 * of expected arguments for the subcommand (including cmd and subcommand
 * names), and a usage message to return if the argument
 * count is outside the expected range.
 */

static cmdDefinition definitions[] = {
    {"new ?name? -mime Mimetype -mode ?FULL, EMBED, BACKGROUND? ?-embed args?",
6 , 9},
    {"destroy pluginName", 3 , 3},
    {"children pluginName", 3 , 3},
    {"getmime ", 2 , 2},
    {"bogus bogusval", 3, 3}
};

/* ----- C++ Comment
// These are the segments that need to be declared in the "C" namespace
// These function names and data will be accessed by programs in the
// Tcl "C" language library.
*/

/*
extern "C" {
*/

/*
 * -----
 *
 * plugextName_Cmd --
 *
 *      plugextName_Cmd is invoked to process the "pluginName" Tcl command.
 *      It will parse a subcommand, and perform the requested action.
 *
 * Results:
 *      A standard Tcl result.
 *
 * Side effects:
 *
 * -----
 */

int plugextName_Cmd(ClientData plugext,
                    Tcl_Interp *interp,
                    int objc,
                    Tcl_Obj *objv[]) {

    /* ClientData plugext;          /* Not used. */
    /* Tcl_Interp *interp;          /* Current interpreter. */

```

```

    /* int objc;                /* Number of arguments. */
    /* Tcl_Obj *CONST objv[];   /* Argument objects. */

int cmdnum;
int result;
Tcl_Obj *returnValue;
CmdReturn *returnStruct;

/*
 * Initialize the return value
 */

returnValue = NULL;
returnStruct = NULL;

/*
 * Check that we have at least a subcommand,
 * else return an Error and the usage message
 */

if (objc < 2) {
    Tcl_WrongNumArgs(interp, 1, objv,
        "subcommand ?fieldName? ?fieldValue?");
    return TCL_ERROR;
}

/*
 * Find this plugext subcommand in the list of subcommands.
 * Tcl_GetIndexFromObj returns the offset of the recognized string,
 * which is used to index into the command definitions table.
 */

result = Tcl_GetIndexFromObj(interp, objv[1], subcommands,
    "subcommand", TCL_EXACT, &cmdnum);

/*
 * If the result is not TCL_OK, then the error message is already
 * in the Tcl Interpreter, this code can immediately return.
 */

if (result != TCL_OK) {
    return TCL_ERROR;
}

/*
 * Check that the argument count matches what's expected for this
 * Subcommand.
 */

if ((objc < definitions[cmdnum].minArgCnt) ||
    (objc > definitions[cmdnum].maxArgCnt) ) {
    Tcl_WrongNumArgs(interp, 1, objv, definitions[cmdnum].usage);
    return TCL_ERROR;
}

result = TCL_OK;

```

```

/*
 * The subcommand is recognized, and has a valid number of arguments
 * Process the command.
 */

switch (cmdnum) {
    case M_new: {
        returnStruct =
            plugextName_newCmd(NULL, interp, objc, objv);
        break;
    }
    case M_destroy: {
        returnStruct =
            plugextName_destroyCmd(NULL, interp, objc, objv);
        break;
    }
    case M_children: {
        returnStruct =
            plugextName_childrenCmd(NULL, interp, objc, objv);
        break;
    }
    case M_getmime: {
        returnStruct =
            plugextName_getmimeCmd(NULL, interp, objc, objv);
        break;
    }
    default: {
        char error[80];
        sprintf(error, "Bad sub-command %s. Has no entry in switch",
            Tcl_GetStringFromObj(objv[1], NULL));
        returnValue = Tcl_NewStringObj(error, -1);
        result = TCL_ERROR;
    }
}

/*
 * Extract an object to return from returnStruct.
 * returnStruct will be NULL if the processing is done in this
 * function and no other function is called.
 */

if (returnStruct != NULL) {
    returnValue = returnStruct->object;
    result = returnStruct->status;
    Tcl_Free ((char *)returnStruct);
}

/*
 * Set the return value and return the status
 */

if (returnValue != NULL) {
    Tcl_SetObjResult(interp, returnValue);
}

debugprt("leaving plugext Cmd\n"); fflush(stdout);

```

```
    return result;
}
/* ----- C++ Comment
// This is the close of the extern "C" section
*/

/*
}
*/
```

```

/*
 * plugextCmdAZ.c --
 * Copyright (c) 1997 Clif Flynt
 * All rights reserved.
 *
 * Permission is hereby granted, without written agreement and without
 * license or royalty fees, to use, copy, modify, and distribute this
 * software and its documentation for any purpose, provided that the
 * above copyright notice and the following two paragraphs appear in
 * all copies of this software.
 *
 * IN NO EVENT SHALL Clif Flynt BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * Clif Flynt SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND Clif Flynt HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
 */
/* Include the usual suspects. */

#ifdef _WIN32
#include <windows.h>
#else
#include "stdlib.h"
#include <stdio.h>
#include "ctype.h"
#endif
#include "plugextInt.h"
#include "extensionLib.h"

static int pluginNameID = 0;

extern Tcl_HashTable *plugext_hashtablePtr;
extern int plugextDebugPrint;

#define PLUGEXTCMDAZ_REVISION "$Revision: 1.14 $"

/*-----
 * CmdReturn *plugextName_newCmd ()--
 * Demonstrates creating a hash entry.
 * Creates a hash entry for Key, with value String
 * Arguments
 *   objv[0]: "plugext"
 *   objv[1]: "load"
 *   objv[2]: fileName
 *   objv[3]: ?IDString?
 *
 * Results
 *   Creates a new hash entry. Sets error if entry already exists.
 */

```

```

* Side Effects:
*   None
-----*/
CmdReturn *plugextName_newCmd (ClientData info,
                               Tcl_Interp *interp,
                               int objc,
                               Tcl_Obj *objv[]) {
    Tcl_HashEntry *hashEntryPtr;
    CmdReturn *returnStructPtr;
    char hashName[90];

    char *hashKeyPtr;
    int isNew;
    NPP_t *NPPhandle;
    PluginData *pluginData;
    InstanceData *instanceData;

    NPError (*NPPNew)(); /* Pointer to the NPP_New function */

    char *pluginName = NULL; /* The name assigned when the plugin was loaded */

    char *mimeType;
    int mode;
    int i, keyvalCount;
    /* Hardcoding array sizes is evil, but I think this will fly. */
    char *tmp, *keyval, *keys[30], *vals[30];
    int status;

    /*
     * Print that the function was called for debugging
     */

    MyPrint("plugextName_newCmd called with %d args : hashName: %s\n", objc,
            Tcl_GetStringFromObj(objv[2], NULL));

    keyvalCount = 0;

    /*
     * Allocate the space and initialize the return structure.
     */

    returnStructPtr = makeCmdReturn();

    /* Confirm that the name is valid (it should be!) and we've got
     * a hash table entry for the plugin Data */

    pluginName = Tcl_GetStringFromObj(objv[0], NULL);

    pluginData = (PluginData *) getHashData(interp, pluginName,
    plugext_hashtablePtr);

    if (pluginData == NULL) {
        setErrorReturns(interp,
            "Plugin named \"%s\" does not exists.\n", pluginName, "", "",
            "Plugin named \"%s\" does not exists.\n", pluginName, "", "",
            "pluginName create", "500");
    }

```

```

        returnStructPtr->status = TCL_ERROR;
        goto done;
    }

    /* If the first arg starts with a "-", its a flag and there is no
     * hardcoded ID name (we choose one). If not, we grab that name.
     */

    tmp = Tcl_GetStringFromObj(objv[2], NULL);

    if (*tmp != '-') {
        strcpy(hashName, Tcl_GetStringFromObj(objv[2], NULL));
    } else {
        sprintf(hashName, "%s%d", pluginName, pluginNameID++);
    }

    /*
     * Extract the string representation of the object argument,
     * and use that as a key into the hash table.
     *
     * If this entry already exists, complain.
     */

    hashEntryPtr = Tcl_CreateHashEntry(plugext_hashtablePtr, hashName, &isNew);

    if (!isNew) {
        setErrorReturns(interp,
            "Plugin named \"%s\" already exists.\n", hashKeyPtr, "", "",
            "plugin with this name exists", "", "", "",
            "pluginName create", "500");

        MyPrint("failed on isNew");

        returnStructPtr->status = TCL_ERROR;
        goto done;
    }

    /*
     * If we are here, then the key is unused.
     * Get the string representation from the object,
     * and make a copy that can be placed into the hash table.
     */

    instanceData = (InstanceData *) Tcl_Alloc (sizeof(InstanceData));
    Tcl_SetHashValue(hashEntryPtr, instanceData);

    Tcl_ListObjAppendElement(interp, pluginData->childList,
        Tcl_NewStringObj(hashName, -1));

    MyPrint("set NPP instanceData: %xx PluginData: %xx\n", instanceData,
        pluginData);

    /* Initialize the function pointers */

    instanceData->netscapeFuncs = &pluginData->netscapeFuncs;
    instanceData->pluginFuncs = &pluginData->pluginFuncs;
    instanceData->pluginReadStream = NULL;

```

```

instanceData->pluginWriteStream = NULL;
instanceData->win                = NULL;
instanceData->readBuffer        = NULL;
instanceData->readLen            = 0;
instanceData->readSize          = 0;
instanceData->readOffset        = 0;
instanceData->putsBuffer        = NULL;
instanceData->streamType        = "No Stream";

instanceData->saved = (NPSavedData *) Tcl_Alloc(sizeof(NPSavedData));
instanceData->saved->len = 0;
instanceData->saved->buf = NULL;
instanceData->interp = interp;

if (returnStructPtr->status != TCL_OK) {goto done;}

/* Initialize these parameters before we check for program settings */

mode = NP_FULL;
mimeType = "text/html";

for (i=0; i<objc; i++) {
    if (!strcmp("-mime", Tcl_GetStringFromObj(objv[i], NULL))) {
        i++;
        mimeType = Tcl_GetStringFromObj(objv[i], NULL);
    }
    if (!strcmp("-mode", Tcl_GetStringFromObj(objv[i], NULL))) {
        i++;
        tmp = Tcl_GetStringFromObj(objv[i], NULL);
        if (!strcmp("FULL", tmp)) {
            mode = NP_FULL;
        }
        if (!strcmp("EMBED", tmp)) {
            mode = NP_EMBED;
        }
        if (!strcmp("BACKGROUND", tmp)) {
            mode = NP_BACKGROUND;
        }
    }
}

if (!strcmp("-embed", Tcl_GetStringFromObj(objv[i], NULL))) {
    i++;
    tmp = Tcl_GetStringFromObj(objv[i], NULL);
    /* tmp == foo=bar baz=abc key="value" */

    keyval = Tcl_Alloc(strlen(tmp)+1);
    strcpy(keyval, tmp);
    tmp = keyval;
    while (*tmp != 0) {
        keys[keyvalCount] = tmp;
        while (*tmp != '=') tmp++;
        *tmp++ = 0;
        if (*tmp == '"') tmp++;
        vals[keyvalCount] = tmp;
        while ((*tmp != ' ') && (*tmp != 0) && (*tmp != '"')) tmp++;
        if (*tmp != 0) {
            *tmp = 0;
            tmp++;
        }
    }
}

```



```

        while (
            ((*tmp == ' ') || (*tmp == '"'))
            && (*tmp != 0)
        ) tmp++;
    }
    keyvalCount++;
}
}

for (i=0; i<keyvalCount; i++) {
    MyPrint("pos: %d key: %s val: %s\n", i, keys[i], vals[i]);
}

NPPhandle = (NPP_t *) Tcl_Alloc (sizeof(NPP_t));

MyPrint("NPPHandle: %xx instanceData/pdata: %xx MODE: %d \n", NPPhandle,
instanceData, mode);

instanceData->handle = NPPhandle;

NPPhandle->ndata = NULL;
NPPhandle->pdata = NULL;

/*
 * KLUDGE - SHOULD ALLOW FOR ALTERNATIVE ARGS TO BE PASSED
 */

{
    NPPluginFuncs *pf;
    pf = instanceData->pluginFuncs;
    NPPNew = pf->newp;
}

if (NPPNew == NULL) {
    char *strErrno;
    strErrno = Tcl_Alloc(99);
    sprintf(strErrno, "%xx", pluginData->dlopenHandle);
    setErrorReturns(interp,
        "Unable to create new plugin. Can't find NPP_New\n", NULL, NULL, NULL,
        "Unable to create new plugin. Can't find NPP_New\n", NULL, NULL, NULL,
        "plugextName_newCmd", strErrno);
    returnStructPtr->status = TCL_ERROR;
    goto done;
}

MyPrint("PRE NPP_New NPPHandle: %xx instanceData %xx Handle->pdata: %xx\n",
NPPhandle, instanceData, NPPhandle->pdata);

status = NPPNew(mimeType, NPPhandle, mode, keyvalCount, keys, vals,
instanceData->saved);

MyPrint("POST NPP_New NPPHandle: %xx instanceData %xx Handle->pdata: %xx\n",
NPPhandle, instanceData, NPPhandle->pdata);
/* status = NPP_New(mimeType, NPPhandle, mode, 0, NULL, NULL, instanceData-
>saved); */

```

```

    if (status != NPERR_NO_ERROR) {
        char err[80];
        sprintf(err, "%d", status);
        Tcl_SetErrorCode(interp, err, (char *) 'NULL');
        sprintf(err, "NPP_New failed on status: %d\n", status);
        Tcl_AddObjErrorInfo(interp, err, strlen(err));
        Tcl_AddErrorInfo(interp, "error in plugext_loadCmd");
        returnStructPtr->status = TCL_ERROR;
    }

    Tcl_CreateObjCommand(interp, hashName,
        (Tcl_ObjCmdProc *) plugInst_Cmd, 0, 0);

done:
    if (returnStructPtr->status == TCL_OK) {
        returnStructPtr->object = Tcl_NewStringObj(hashName, -1);
    } else {
        returnStructPtr->object =
            Tcl_NewStringObj("Failed to create new plugin instance", -1);
    }

    MyPrint("Leaving loadCmd \n"); fflush(stdout);

    return returnStructPtr;
}

/*-----
 * CmdReturn *plugextName_childrenCmd ()--
 *   Demonstrates creating a hash entry.
 *   Creates a hash entry for Key, with value String
 * Arguments
 *   objv[0]:      "plugext"
 *   objv[1]:      "load"
 *   objv[2]:      fileName
 *   objv[3]:      ?IDString?
 *
 * Results
 *   Creates a new hash entry.  Sets error if entry already exists.
 *
 * Side Effects:
 *   None
 *-----*/
CmdReturn *plugextName_childrenCmd (ClientData info,
    Tcl_Interp *interp,
    int objc,
    Tcl_Obj *objv[]) {

    CmdReturn *returnStructPtr;

    PluginData *pluginData;
    char *pluginName = NULL; /* The name assigned when the plugin was loaded */

    /*
     * Print that the function was called for debugging
     */

```

```

MyPrint("plugextName_childrenCmd called with %d args : hashName: %s\n",
        objc, Tcl_GetStringFromObj(objv[2], NULL));

/*
 * Allocate the space and initialize the return structure.
 */

returnStructPtr = makeCmdReturn();

/* Confirm that the name is valid (it should be!) and we've got
 * a hash table entry for the plugin Data */

pluginName = Tcl_GetStringFromObj(objv[0], NULL);

pluginData = (PluginData *) getHashData(interp, pluginName,
        plugext_hashtablePtr);

if (pluginData == NULL) {
    returnStructPtr->status = TCL_ERROR;
    goto done;
}

returnStructPtr->object = pluginData->childList;

done:

MyPrint("Leaving loadCmd \n"); fflush(stdout);

return returnStructPtr;
}

/*-----
 * CmdReturn *plugextName_getmimeCmd ()--
 *   Demonstrates creating a hash entry.
 *   Creates a hash entry for Key, with value String
 * Arguments
 *   objv[0]:      "plugext"
 *   objv[1]:      "load"
 *   objv[2]:      fileName
 *   objv[3]:      ?IDString?
 *
 * Results
 *   Creates a new hash entry.  Sets error if entry already exists.
 *
 * Side Effects:
 *   None
 *-----*/
CmdReturn *plugextName_getmimeCmd (ClientData info,
        Tcl_Interp *interp,
        int objc,
        Tcl_Obj *objv[]) {

    CmdReturn *returnStructPtr;

```

```

PluginData *pluginData;
char *pluginName = NULL; /* The name assigned when the plugin was loaded */
char *mimeString;
char * (*NPPMimeDesc)(); /* Pointer to the NPP_Mime description function */

char *tmpMIME;
tmpMIME = Tcl_Alloc(99);

/*
 * Print that the function was called for debugging
 */

MyPrint("plugextName_getmimeCmd called with %d args : hashName: %s\n",
        objc, Tcl_GetStringFromObj(objv[2], NULL));

/*
 * Allocate the space and initialize the return structure.
 */

returnStructPtr = makeCmdReturn();

/* Confirm that the name is valid (it should be!) and we've got
 * a hash table entry for the plugin Data */

pluginName = Tcl_GetStringFromObj(objv[0], NULL);

pluginData = (PluginData *) getHashData(interp, pluginName,
        plugext_hashtablePtr);

if (pluginData == NULL) {
    returnStructPtr->status = TCL_ERROR;
    goto done;
}

#if defined (_WIN32)
{
    DWORD dwLen;
    char name[255];

    BOOL bRet;
    char *value;
    int len;

    struct {
        WORD    first;
        WORD    second;
    } *lpBuffer;

    VerQueryValue(pluginData->VersionData, "\\VarFileInfo\\Translation",
        (LPVOID*) &lpBuffer, (unsigned int *) &dwLen);

    if (dwLen == 0) {
        setErrorReturns(interp,
            "Failed to get second dwLen", NULL, NULL, NULL,
            "Failed to get second dwLen", NULL, NULL, NULL,

```

```

        "plugextName_getmimeCmd", "516");
    returnStructPtr->status = TCL_ERROR;
    goto done;
}

wsprintf(name, "\\StringFileInfo\\%04x%04x\\MIMEType",
        lpBuffer->first, lpBuffer->second);

bRet = VerQueryValue(pluginData->VersionData, name, &value, &len);

if (!bRet) {
    setErrorReturns(interp,
        "Failed to get MIMEType", NULL, NULL, NULL,
        "Failed to get MIMEType", NULL, NULL, NULL,
        "plugextName_getmimeCmd", "517");
    returnStructPtr->status = TCL_ERROR;
    goto done;
}

mimeString = value;
}
#else
    NPPMimeDesc = dlsym(pluginData->dlopenHandle, "NPP_GetMIMEDescription");
    if (NPPMimeDesc == NULL) {
        NPPMimeDesc = dlsym(pluginData->dlopenHandle, "NP_GetMIMEDescription");
    }
    if (NPPMimeDesc == NULL) {
        setErrorReturns(interp,
            "Unable to determine MIDI type. Can't find NP_GetMIMEDescription",
            NULL, NULL, NULL,
            "Unable to determine MIDI type. Can't find NP_GetMIMEDescription",
            NULL, NULL, NULL,
            "plugextName_getmimeCmd", "513");
        returnStructPtr->status = TCL_ERROR;
    }

    if (NPPMimeDesc == NULL) {
        mimeString = "Unknown/unknown";
    } else {
        mimeString = NPPMimeDesc();
    }
}

#endif

returnStructPtr->object = Tcl_NewStringObj(mimeString, -1);

done:

MyPrint("Leaving loadCmd \n"); fflush(stdout);

return returnStructPtr;
}

```

```

/*-----

```

```

* CmdReturn *plugextName_destroyCmd ()--
*   Demonstrates creating a hash entry.
*   Creates a hash entry for Key, with value String
* Arguments
*   objv[0]:      "plugext"
*   objv[1]:      "load"
*   objv[2]:      fileName
*   objv[3]:      ?IDString?
*
* Results
*   Creates a new hash entry.  Sets error if entry already exists.
*
* Side Effects:
*   None
-----*/
CmdReturn *plugextName_destroyCmd (ClientData info,
                                   Tcl_Interp *interp,
                                   int objc,
                                   Tcl_Obj *objv[]) {

    CmdReturn *returnStructPtr;

    PluginData *pluginData;
    char *pluginName = NULL; /* The name assigned when the plugin was loaded */

    /*
     * Print that the function was called for debugging
     */

    MyPrint("plugextName_destroyCmd called with %d args : hashName: %s\n",
            objc, Tcl_GetStringFromObj(objv[2], NULL));

    /*
     * Allocate the space and initialize the return structure.
     */

    returnStructPtr = makeCmdReturn();

    /* Confirm that the name is valid (it should be!) and we've got
     * a hash table entry for the plugin Data */

    pluginName = Tcl_GetStringFromObj(objv[0], NULL);

    pluginData = (PluginData *) getHashData(interp, pluginName,
        plugext_hashtablePtr);

    if (pluginData == NULL) {
        returnStructPtr->status = TCL_ERROR;
        goto done;
    }

    returnStructPtr->object = Tcl_NewStringObj("NOT IMPLEMENTED YET", -1);
    returnStructPtr->status = TCL_ERROR;

done:

    MyPrint("Leaving loadCmd \n"); fflush(stdout);

```

```
return returnStructPtr;
```

```
}
```

```

/*
 * plugextInt.h --
 *
 *      Declarations used by the plugexttcl extension
 *
 * Copyright (c) 1997 Clif Flynt
 *
 * See the file "license.terms" for information on usage and
 * redistribution of this file, and for a
 * DISCLAIMER OF ALL WARRANTIES.
 *
 * RCS: $Log:
 */

#ifndef _PLUGEXTINT
#define _PLUGEXTINT

/*
 * Declare the #includes that will be used by this extension
 */

#include "extensionLib.h"

/* typedef struct Tcl_Obj Tcl_Obj; */

#ifdef __cplusplus
extern "C" {
#endif

#include "tcl805.h"

#ifdef __cplusplus
}
#endif

#include <string.h>
#include "npupp.h"
#include "npapi.h"
/* #include <sys/utsname.h> */
/* #include <iostream.h> */

/*
 * Define the major and minor version numbers.
 * Note: VERSION is defined as a string, not integers.
 *      MAJOR and MINOR are defined as integers.
 */

#define PLUGEXT_VERSION "1.0"
#define PLUGEXT_MAJOR_VERSION 1
#define PLUGEXT_MINOR_VERSION 0

/* MAY BE UNNECESSARY - APPEARS IN DOCUMENTATION, but NOT npapi.h */

#define NP_BACKGROUND 3

typedef struct pluginData {

```



```

#if defined (_WIN32)
    HINSTANCE dlopenHandle;
#else
    void *dlopenHandle;
#endif
    NPNetscapeFuncs netscapeFuncs;
    NPPluginFuncs pluginFuncs;
    Tcl_Obj *childList;
    char *DLLname;          /* The name of this plugin (xx.dll/xx.so) */
#if defined (_WIN32)
    void *VersionData;      /* To be queried as necessary */
#endif
} PluginData;

typedef struct instanceData {
    NPNetscapeFuncs *netscapeFuncs;
    NPPluginFuncs *pluginFuncs;
    NPStream *pluginReadStream;
    NPStream *pluginWriteStream;
    NPP handle;
    NPWindow *win;
    NPSavedData *saved;
    char *streamType;       /* asfile, normal, asfileonly */
    Tcl_Channel tclReadChannel;
    Tcl_Channel tclPutsChannel;
    unsigned char *readBuffer;
    unsigned char *putsBuffer;
    int readLen;            /* How much data is in readBuffer */
    int readOffset;         /* Where does it start */
    int readSize;           /* How much space is there */
    int maxAccept;          /* The first WriteReady return */
    Tcl_Interp *interp;     /* For use by the I/O routines */
} InstanceData;

/*
 * VC++ has an alternate entry point called DllMain, so we
 * need to rename our entry point.
 */

#if defined(__WIN32__)
#   define WIN32_LEAN_AND_MEAN
#   include <windows.h>
#   undef WIN32_LEAN_AND_MEAN
#   if defined(_MSC_VER)
#       define EXPORT(a,b) __declspec(dllexport) a b
#       define DllEntryPoint DllMain
#   else
#       if defined(__BORLANDC__)
#           define EXPORT(a,b) a _export b
#       else
#           define EXPORT(a,b) a b
#       endif
#   endif
#else
#   define EXPORT(a,b) a b
#endif

```

```

/*
 * Function Prototypes for the commands that actually do the
 * processing.
 * Two macros are used in these prototypes:
 *
 * EXTERN_EXPORT is for functions that must interact with the
 * Microsoft or Borland C++ DLL loader.
 * ANSI_ARGS is defined in tcl.h
 * ANSI_ARGS returns an empty string for non-ANSI C
 * compilers, and returns it's arguments for ANSI C
 * compilers.
 */

/**
extern "C" {
***/
/*****
// EXTERN_EXPORT(int,plugext_AppInit) _ANSI_ARGS_ ((Tcl_Interp *));
// EXTERN_EXPORT(int,plugext_Init) _ANSI_ARGS_ ((Tcl_Interp *));
// EXTERN_EXPORT(int,plugext_Cmd) _ANSI_ARGS_ ((ClientData, Tcl_Interp *, int,
Tcl_Obj **));
*****/

int plugext_AppInit _ANSI_ARGS_ ((Tcl_Interp *));
int plugext_Init _ANSI_ARGS_ ((Tcl_Interp *));
int plugext_Cmd _ANSI_ARGS_ ((ClientData, Tcl_Interp *, int, Tcl_Obj **));

/**
}
***/

int plugext_loadInit _ANSI_ARGS_ ((Tcl_Interp *, char *));

int plugextName_Cmd _ANSI_ARGS_ ((ClientData,
Tcl_Interp *, int, Tcl_Obj **));

int plugInst_Cmd _ANSI_ARGS_ ((ClientData,
Tcl_Interp *, int, Tcl_Obj **));

CmdReturn *plugext_createCmd _ANSI_ARGS_ ((ClientData,
Tcl_Interp *, int, Tcl_Obj **));

CmdReturn *plugextName_newCmd _ANSI_ARGS_ ((ClientData,
Tcl_Interp *, int, Tcl_Obj **));
CmdReturn *plugextName_destroyCmd _ANSI_ARGS_ ((ClientData,
Tcl_Interp *, int, Tcl_Obj **));
CmdReturn *plugextName_childrenCmd _ANSI_ARGS_ ((ClientData,
Tcl_Interp *, int, Tcl_Obj **));
CmdReturn *plugextName_getmimeCmd _ANSI_ARGS_ ((ClientData,
Tcl_Interp *, int, Tcl_Obj **));

CmdReturn *plugInst_openCmd _ANSI_ARGS_ ((ClientData,
Tcl_Interp *, int, Tcl_Obj **));

```

```

CmdReturn *plugInst_destroyCmd _ANSI_ARGS_ ((ClientData,
    Tcl_Interp *, int, Tcl_Obj **));
CmdReturn *plugInst_closeCmd _ANSI_ARGS_ ((ClientData,
    Tcl_Interp *, int, Tcl_Obj **));
CmdReturn *plugInst_setwindowCmd _ANSI_ARGS_ ((ClientData,
    Tcl_Interp *, int, Tcl_Obj **));
CmdReturn *plugInst_writereadyCmd _ANSI_ARGS_ ((ClientData,
    Tcl_Interp *, int, Tcl_Obj **));
CmdReturn *plugInst_streamfileCmd _ANSI_ARGS_ ((ClientData,
    Tcl_Interp *, int, Tcl_Obj **));
CmdReturn *plugInst_streamTypeCmd _ANSI_ARGS_ ((ClientData,
    Tcl_Interp *, int, Tcl_Obj **));

```

```

Tcl_ChannelType *makeChannelTypeStructure();

```

```

CmdReturn *plugext_ArraystrCmd _ANSI_ARGS_ ((ClientData,
    Tcl_Interp *, int, Tcl_Obj **));
CmdReturn *plugext_GetstrCmd _ANSI_ARGS_ ((ClientData,
    Tcl_Interp *, int, Tcl_Obj **));
void plugext_InitHashTable _ANSI_ARGS_ (());

```

```

/*
 * For debugging printf's.
 */

```

```

#define debugprt if (plugextDebugPrint>0) printf

```

```

#endif
/* End _PLUGEXTINT */

```

```

/*
 * plugextCmd.c --
 * Copyright (c) 1997 Clif Flynt
 * All rights reserved.
 *
 * Permission is hereby granted, without written agreement and without
 * license or royalty fees, to use, copy, modify, and distribute this
 * software and its documentation for any purpose, provided that the
 * above copyright notice and the following two paragraphs appear in
 * all copies of this software.
 *
 * IN NO EVENT SHALL Clif Flynt BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * Clif Flynt SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND Clif Flynt HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
 */
/* Include the usual suspects. */

#if defined (_WIN32)
#include <windows.h>
#endif
#include "stdlib.h"
#include "ctype.h"
#include "plugextInt.h"

#define plugInst_REVISION "$Revision: 1.5 $"

extern int plugextDebugPrint;

/*
 * Define the sub commands
 *
 * These strings define the subcommands that the plugext command
 * supports.
 *
 * To add a new subcommand, add the new subcommand string,
 * #define and entry in cmdDefinition
 *
 * Note: Order is important.
 *
 * These are the subcommands that will be recognized
 */

static char *subcommands[] = {
    "open", "destroy", "close", "setwindow", "writeready", "streamfile",
    "streamtype", "sendFile", "bogus", NULL};

/*
 * These #defines define the positions of the subcommands.
 * You can use enums if you are certain your compiler will provide the

```

```

* same numbers as this.
*/

#define M_open 0
#define M_destroy 1
#define M_close 2
#define M_setwindow 3
#define M_writeready 4
#define M_streamfile 5
#define M_streamtype 6
#define M_sendFile 7
#define M_bogus 8

/*
* The cmdDefinition structure describes the minimum and maximum number
* of expected arguments for the subcommand (including cmd and subcommand
* names), and a usage message to return if the argument
* count is outside the expected range.
*/

static cmdDefinition definitions[] = {
    {"open ?-mime Mimetype? -seekable ?FULL, EMBED, BACKGROUND? 'r/w'", 3 , 7},
    {"destroy ", 2 , 2},
    {"close channelId", 3 , 3},
    {"setwindow windowName", 3 , 3},
    {"writeready channel", 3 , 3},
    {"streamfile fileName", 3 , 3},
    {"streamtype", 2 , 2},
    {"sendFile fileName", 3 , 3},
    {"bogus bogusval", 3 , 3}
};

/* ----- C++ Comment
// These are the segments that need to be declared in the "C" namespace
// These function names and data will be accessed by programs in the
// Tcl "C" language library.
*/

/*
extern "C" {
*/

/*
* -----
*
* plugInst_Cmd --
*
*     plugextName_Cmd is invoked to process the "pluginName" Tcl command.
*     It will parse a subcommand, and perform the requested action.
*
* Results:
*     A standard Tcl result.
*
* Side effects:
* -----
*/

```

```

int plugInst_Cmd(ClientData plugext,
                  Tcl_Interp *interp,
                  int objc,
                  Tcl_Obj *objv[]) {

    /* ClientData plugext;          /* Not used. */
    /* Tcl_Interp *interp;          /* Current interpreter. */
    /* int objc;                    /* Number of arguments. */
    /* Tcl_Obj *CONST objv[];       /* Argument objects. */

    int cmdnum;
    int result;
    Tcl_Obj *returnValue;
    CmdReturn *returnStruct;

    /*
     * Initialize the return value
     */

    returnValue = NULL;
    returnStruct = NULL;

    /*
     * Check that we have at least a subcommand,
     * else return an Error and the usage message
     */

    if (objc < 2) {
        Tcl_WrongNumArgs(interp, 1, objv,
                          "subcommand ?fieldName? ?fieldValue?");
        return TCL_ERROR;
    }

    /*
     * Find this plugext subcommand in the list of subcommands.
     * Tcl_GetIndexFromObj returns the offset of the recognized string,
     * which is used to index into the command definitions table.
     */

    result = Tcl_GetIndexFromObj(interp, objv[1], subcommands,
                                  "subcommand", TCL_EXACT, &cmdnum);

    /*
     * If the result is not TCL_OK, then the error message is already
     * in the Tcl Interpreter, this code can immediately return.
     */

    if (result != TCL_OK) {
        return TCL_ERROR;
    }

    /*
     * Check that the argument count matches what's expected for this
     * Subcommand.
     */

```

```

if ((objc < definitions[cmdnum].minArgCnt) ||
    (objc > definitions[cmdnum].maxArgCnt) ) {
    Tcl_WrongNumArgs(interp, 1, objv, definitions[cmdnum].usage);
    return TCL_ERROR;
}

result = TCL_OK;

/*
 * The subcommand is recognized, and has a valid number of arguments
 * Process the command.
 */

switch (cmdnum) {
    case M_open: {
        returnStruct =
            plugInst_openCmd(NULL, interp, objc, objv);
        break;
    }
    case M_destroy: {
        returnStruct =
            plugInst_destroyCmd(NULL, interp, objc, objv);
        break;
    }
    case M_close: {
        returnStruct =
            plugInst_closeCmd(NULL, interp, objc, objv);
        break;
    }
    case M_setwindow: {
        returnStruct =
            plugInst_setwindowCmd(NULL, interp, objc, objv);
        break;
    }
    case M_writeready: {
        returnStruct =
            plugInst_writereadyCmd(NULL, interp, objc, objv);
        break;
    }
    case M_streamfile: {
        returnStruct =
            plugInst_streamfileCmd(NULL, interp, objc, objv);
        break;
    }
    case M_streamtype: {
        returnStruct =
            plugInst_streamTypeCmd(NULL, interp, objc, objv);
        break;
    }
    case M_sendFile: {
        returnStruct =
            plugInst_sendFileCmd(NULL, interp, objc, objv);
        break;
    }
    default: {
        char error[80];

```

```

        sprintf(error, "Bad sub-command %s. Has no entry in switch",
            Tcl_GetStringFromObj(objv[1], NULL));
        returnValue = Tcl_NewStringObj(error, -1);
        result = TCL_ERROR;
    }
}

/*
 * Extract an object to return from returnStruc.
 * returnStruc will be NULL if the processing is done in this
 * function and no other function is called.
 */

if (returnStruc != NULL) {
    returnValue = returnStruc->object;
    result = returnStruc->status;
    Tcl_Free ((char *)returnStruc);
}

/*
 * Set the return value and return the status
 */

if (returnValue != NULL) {
    Tcl_SetObjResult(interp, returnValue);
}

    debugprt("leaving plugextInst Cmd\n"); fflush(stdout);

    return result;
}
/* ----- C++ Comment
// This is the close of the extern "C" section
*/

/*
}
*/

```



```

/*
 * tcl.h --
 *
 * This header file describes the externally-visible facilities
 * of the Tcl interpreter.
 *
 * Copyright (c) 1987-1994 The Regents of the University of California.
 * Copyright (c) 1994-1997 Sun Microsystems, Inc.
 * Copyright (c) 1993-1996 Lucent Technologies.
 * Copyright (c) 1998-1999 Scriptics Corporation.
 *
 * See the file "license.terms" for information on usage and redistribution
 * of this file, and for a DISCLAIMER OF ALL WARRANTIES.
 *
 * RCS: @(#) $Id: tcl805.h,v 1.1 1999/08/01 14:58:49 cliff Exp $
 */

```

```

#ifndef _TCL
#define _TCL

```

```

/*
 * When version numbers change here, must also go into the following files
 * and update the version numbers:
 *
 * README
 * library/init.tcl      (only if major.minor changes, not patchlevel)
 * unix/configure.in
 * win/makefile.bc       (only if major.minor changes, not patchlevel)
 * win/makefile.vc       (only if major.minor changes, not patchlevel)
 * win/README
 * win/README.binary
 * mac/README
 *
 * The release level should be 0 for alpha, 1 for beta, and 2 for
 * final/patch. The release serial value is the number that follows the
 * "a", "b", or "p" in the patch level; for example, if the patch level
 * is 7.6b2, TCL_RELEASE_SERIAL is 2. It restarts at 1 whenever the
 * release level is changed, except for the final release which is 0
 * (the first patch will start at 1).
 */

```

```

#define TCL_MAJOR_VERSION 8
#define TCL_MINOR_VERSION 0
#define TCL_RELEASE_LEVEL 2
#define TCL_RELEASE_SERIAL 5

#define TCL_VERSION "8.0"
#define TCL_PATCH_LEVEL "8.0.5"

```

```

/*
 * The following definitions set up the proper options for Windows
 * compilers. We use this method because there is no autoconf equivalent.
 */

```

```

#ifndef __WIN32__
# if defined(_WIN32) || defined(WIN32)
#   define __WIN32__

```

```

# endif
#endif

#ifdef __WIN32__
#   ifndef STRICT
#       define STRICT
#   endif
#   ifndef USE_PROTOTYPE
#       define USE_PROTOTYPE 1
#   endif
#   ifndef HAS_STDARG
#       define HAS_STDARG 1
#   endif
#   ifndef USE_PROTOTYPE
#       define USE_PROTOTYPE 1
#   endif

/*
 * Under Windows we need to call Tcl_Alloc in all cases to avoid competing
 * C run-time library issues.
 */

#   ifndef USE_TCLALLOC
#       define USE_TCLALLOC 1
#   endif
#endif /* __WIN32__ */

/*
 * The following definitions set up the proper options for Macintosh
 * compilers. We use this method because there is no autoconf equivalent.
 */

#ifdef MAC_TCL
#   ifndef HAS_STDARG
#       define HAS_STDARG 1
#   endif
#   ifndef USE_TCLALLOC
#       define USE_TCLALLOC 1
#   endif
#   ifndef NO_STRERROR
#       define NO_STRERROR 1
#   endif
#endif

/*
 * Utility macros: STRINGIFY takes an argument and wraps it in "" (double
 * quotation marks), JOIN joins two arguments.
 */

#define VERBATIM(x) x
#ifdef _MSC_VER
#   define STRINGIFY(x) STRINGIFY1(x)
#   define STRINGIFY1(x) #x
#   define JOIN(a,b) JOIN1(a,b)
#   define JOIN1(a,b) a##b
#else
#   ifdef RESOURCE_INCLUDED

```

```

# define STRINGIFY(x) STRINGIFY1(x)
# define STRINGIFY1(x) #x
# define JOIN(a,b) JOIN1(a,b)
# define JOIN1(a,b) a##b
# else
# ifdef __STDC__
#   define STRINGIFY(x) #x
#   define JOIN(a,b) a##b
# else
#   define STRINGIFY(x) "x"
#   define JOIN(a,b) VERBATIM(a)VERBATIM(b)
# endif
# endif
#endif

/*
 * A special definition used to allow this header file to be included
 * in resource files so that they can get obtain version information from
 * this file. Resource compilers don't like all the C stuff, like typedefs
 * and procedure declarations, that occur below.
 */

#ifndef RESOURCE_INCLUDED

#ifndef BUFSIZ
#include <stdio.h>
#endif

/*
 * Definitions that allow Tcl functions with variable numbers of
 * arguments to be used with either varargs.h or stdarg.h.  TCL_VARARGS
 * is used in procedure prototypes.  TCL_VARARGS_DEF is used to declare
 * the arguments in a function definition: it takes the type and name of
 * the first argument and supplies the appropriate argument declaration
 * string for use in the function definition.  TCL_VARARGS_START
 * initializes the va_list data structure and returns the first argument.
 */

#if defined(__STDC__) || defined(HAS_STDARG)
#   define TCL_VARARGS(type, name) (type name, ...)
#   define TCL_VARARGS_DEF(type, name) (type name, ...)
#   define TCL_VARARGS_START(type, name, list) (va_start(list, name), name)
#else
#   ifdef __cplusplus
#       define TCL_VARARGS(type, name) (type name, ...)
#       define TCL_VARARGS_DEF(type, name) (type va_alist, ...)
#   else
#       define TCL_VARARGS(type, name) ()
#       define TCL_VARARGS_DEF(type, name) (va_alist)
#   endif
#   define TCL_VARARGS_START(type, name, list) \
        (va_start(list), va_arg(list, type))
#endif

/*
 * Macros used to declare a function to be exported by a DLL.
 * Used by Windows, maps to no-op declarations on non-Windows systems.

```

```

* The default build on windows is for a DLL, which causes the DLLIMPORT
* and DLLEXPORT macros to be nonempty. To build a static library, the
* macro STATIC_BUILD should be defined.
* The support follows the convention that a macro called BUILD_xxxx, where
* xxxx is the name of a library we are building, is set on the compile line
* for sources that are to be placed in the library. See BUILD_tcl in this
* file for an example of how the macro is to be used.
*/

```

```

#ifdef __WIN32__
# ifdef STATIC_BUILD
#   define DLLIMPORT
#   define DLLEXPORT
# else
#   if defined(_MSC_VER) || (defined(__GNUC__) && defined(__declspec))
#     define DLLIMPORT __declspec(dllimport)
#     define DLLEXPORT __declspec(dllexport)
#   else
#     define DLLIMPORT
#     define DLLEXPORT
#   endif
# endif
#else
# define DLLIMPORT
# define DLLEXPORT
#endif

```

```

#ifdef TCL_STORAGE_CLASS
# undef TCL_STORAGE_CLASS
#endif
#ifdef BUILD_tcl
# define TCL_STORAGE_CLASS DLLEXPORT
#else
# define TCL_STORAGE_CLASS DLLIMPORT
#endif

```

```

/*
* Definitions that allow this header file to be used either with or
* without ANSI C features like function prototypes.
*/

```

```

#undef _ANSI_ARGS_
#undef CONST

```

```

#if ((defined(__STDC__) || defined(SABER)) && !defined(NO_PROTOTYPE)) ||
defined(__cplusplus) || defined(USE_PROTOTYPE)
#   define _USING_PROTOTYPES_ 1
#   define _ANSI_ARGS_(x)      x
#   define CONST const
#else
#   define _ANSI_ARGS_(x)      ()
#   define CONST
#endif

```

```

#ifdef __cplusplus
#   define EXTERN extern "C" TCL_STORAGE_CLASS
#else

```

```

#   define EXTERN extern TCL_STORAGE_CLASS
#endif

/*
 * Macro to use instead of "void" for arguments that must have
 * type "void *" in ANSI C; maps them to type "char *" in
 * non-ANSI systems.
 */
#ifndef __WIN32__
#ifndef VOID
#   ifdef __STDC__
#       define VOID void
#   else
#       define VOID char
#   endif
#endif
#else /* __WIN32__ */
/*
 * The following code is copied from winnt.h
 */
#ifndef VOID
#define VOID void
typedef char CHAR;
typedef short SHORT;
typedef long LONG;
#endif
#endif /* __WIN32__ */

/*
 * Miscellaneous declarations.
 */

#ifndef NULL
#define NULL 0
#endif

#ifndef _CLIENTDATA
#   if defined(__STDC__) || defined(__cplusplus)
typedef void *ClientData;
#   else
typedef int *ClientData;
#   endif /* __STDC__ */
#define _CLIENTDATA
#endif

/*
 * Data structures defined opaquely in this module. The definitions below
 * just provide dummy types. A few fields are made visible in Tcl_Interp
 * structures, namely those used for returning a string result from
 * commands. Direct access to the result field is discouraged in Tcl 8.0.
 * The interpreter result is either an object or a string, and the two
 * values are kept consistent unless some C code sets interp->result
 * directly. Programmers should use either the procedure Tcl_GetObjResult()
 * or Tcl_GetStringResult() to read the interpreter's result. See the
 * SetResult man page for details.
 *
 * Note: any change to the Tcl_Interp definition below must be mirrored

```

```

* in the "real" definition in tclInt.h.
*
* Note: Tcl_ObjCmdProc procedures do not directly set result and freeProc.
* Instead, they set a Tcl_Obj member in the "real" structure that can be
* accessed with Tcl_GetObjResult() and Tcl_SetObjResult().
*/

typedef struct Tcl_Interp {
    char *result;          /* If the last command returned a string
                           * result, this points to it. */
    void (*freeProc) _ANSI_ARGS_((char *blockPtr));
                           /* Zero means the string result is
                           * statically allocated. TCL_DYNAMIC means
                           * it was allocated with ckalloc and should
                           * be freed with ckfree. Other values give
                           * the address of procedure to invoke to
                           * free the result. Tcl_Eval must free it
                           * before executing next command. */
    int errorLine;         /* When TCL_ERROR is returned, this gives
                           * the line number within the command where
                           * the error occurred (1 if first line). */
} Tcl_Interp;

typedef struct Tcl_AsyncHandler_ *Tcl_AsyncHandler;
typedef struct Tcl_Channel_ *Tcl_Channel;
typedef struct Tcl_Command_ *Tcl_Command;
typedef struct Tcl_Event Tcl_Event;
typedef struct Tcl_Pid_ *Tcl_Pid;
typedef struct Tcl_RegExp_ *Tcl_RegExp;
typedef struct Tcl_TimerToken_ *Tcl_TimerToken;
typedef struct Tcl_Trace_ *Tcl_Trace;
typedef struct Tcl_Var_ *Tcl_Var;

/*
* When a TCL command returns, the interpreter contains a result from the
* command. Programmers are strongly encouraged to use one of the
* procedures Tcl_GetObjResult() or Tcl_GetStringResult() to read the
* interpreter's result. See the SetResult man page for details. Besides
* this result, the command procedure returns an integer code, which is
* one of the following:
*
* TCL_OK           Command completed normally; the interpreter's
*                   result contains the command's result.
* TCL_ERROR        The command couldn't be completed successfully;
*                   the interpreter's result describes what went wrong.
* TCL_RETURN       The command requests that the current procedure
*                   return; the interpreter's result contains the
*                   procedure's return value.
* TCL_BREAK        The command requests that the innermost loop
*                   be exited; the interpreter's result is meaningless.
* TCL_CONTINUE     Go on to the next iteration of the current loop;
*                   the interpreter's result is meaningless.
*/

#define TCL_OK          0
#define TCL_ERROR      1
#define TCL_RETURN     2

```

```

#define TCL_BREAK 3
#define TCL_CONTINUE 4

#define TCL_RESULT_SIZE 200

/*
 * Argument descriptors for math function callbacks in expressions:
 */

typedef enum {TCL_INT, TCL_DOUBLE, TCL_EITHER} Tcl_ValueType;
typedef struct Tcl_Value {
    Tcl_ValueType type;          /* Indicates intValue or doubleValue is
                                * valid, or both. */
    long intValue;              /* Integer value. */
    double doubleValue;         /* Double-precision floating value. */
} Tcl_Value;

/*
 * Forward declaration of Tcl_Obj to prevent an error when the forward
 * reference to Tcl_Obj is encountered in the procedure types declared
 * below.
 */

struct Tcl_Obj;

/*
 * Procedure types defined by Tcl:
 */

typedef int (Tcl_AppInitProc) _ANSI_ARGS_((Tcl_Interp *interp));
typedef int (Tcl_AsyncProc) _ANSI_ARGS_((ClientData clientData,
    Tcl_Interp *interp, int code));
typedef void (Tcl_ChannelProc) _ANSI_ARGS_((ClientData clientData, int mask));
typedef void (Tcl_CloseProc) _ANSI_ARGS_((ClientData data));
typedef void (Tcl_CmdDeleteProc) _ANSI_ARGS_((ClientData clientData));
typedef int (Tcl_CmdProc) _ANSI_ARGS_((ClientData clientData,
    Tcl_Interp *interp, int argc, char *argv[]));
typedef void (Tcl_CmdTraceProc) _ANSI_ARGS_((ClientData clientData,
    Tcl_Interp *interp, int level, char *command, Tcl_CmdProc *proc,
    ClientData cmdClientData, int argc, char *argv[]));
typedef void (Tcl_DupInternalRepProc) _ANSI_ARGS_((struct Tcl_Obj *srcPtr,
    struct Tcl_Obj *dupPtr));
typedef int (Tcl_EventProc) _ANSI_ARGS_((Tcl_Event *evPtr, int flags));
typedef void (Tcl_EventCheckProc) _ANSI_ARGS_((ClientData clientData,
    int flags));
typedef int (Tcl_EventDeleteProc) _ANSI_ARGS_((Tcl_Event *evPtr,
    ClientData clientData));
typedef void (Tcl_EventSetupProc) _ANSI_ARGS_((ClientData clientData,
    int flags));
typedef void (Tcl_ExitProc) _ANSI_ARGS_((ClientData clientData));
typedef void (Tcl_FileProc) _ANSI_ARGS_((ClientData clientData, int mask));
typedef void (Tcl_FileFreeProc) _ANSI_ARGS_((ClientData clientData));
typedef void (Tcl_FreeInternalRepProc) _ANSI_ARGS_((struct Tcl_Obj *objPtr));
typedef void (Tcl_FreeProc) _ANSI_ARGS_((char *blockPtr));
typedef void (Tcl_IdleProc) _ANSI_ARGS_((ClientData clientData));
typedef void (Tcl_InterpDeleteProc) _ANSI_ARGS_((ClientData clientData,
    Tcl_Interp *interp));

```

```

typedef int (Tcl_MathProc) _ANSI_ARGS_((ClientData clientData,
    Tcl_Interp *interp, Tcl_Value *args, Tcl_Value *resultPtr));
typedef void (Tcl_NamespaceDeleteProc) _ANSI_ARGS_((ClientData clientData));
typedef int (Tcl_ObjCmdProc) _ANSI_ARGS_((ClientData clientData,
    Tcl_Interp *interp, int objc, struct Tcl_Obj * CONST objv[]));
typedef int (Tcl_PackageInitProc) _ANSI_ARGS_((Tcl_Interp *interp));
typedef void (Tcl_TcpAcceptProc) _ANSI_ARGS_((ClientData callbackData,
    Tcl_Channel chan, char *address, int port));
typedef void (Tcl_TimerProc) _ANSI_ARGS_((ClientData clientData));
typedef int (Tcl_SetFromAnyProc) _ANSI_ARGS_((Tcl_Interp *interp,
    struct Tcl_Obj *objPtr));
typedef void (Tcl_UpdateStringProc) _ANSI_ARGS_((struct Tcl_Obj *objPtr));
typedef char *(Tcl_VarTraceProc) _ANSI_ARGS_((ClientData clientData,
    Tcl_Interp *interp, char *part1, char *part2, int flags));

```

```

/*
 * The following structure represents a type of object, which is a
 * particular internal representation for an object plus a set of
 * procedures that provide standard operations on objects of that type.
 */

```

```

typedef struct Tcl_ObjType {
    char *name; /* Name of the type, e.g. "int". */
    Tcl_FreeInternalRepProc *freeIntRepProc;
    /* Called to free any storage for the type's
     * internal rep. NULL if the internal rep
     * does not need freeing. */
    Tcl_DupInternalRepProc *dupIntRepProc;
    /* Called to create a new object as a copy
     * of an existing object. */
    Tcl_UpdateStringProc *updateStringProc;
    /* Called to update the string rep from the
     * type's internal representation. */
    Tcl_SetFromAnyProc *setFromAnyProc;
    /* Called to convert the object's internal
     * rep to this type. Frees the internal rep
     * of the old type. Returns TCL_ERROR on
     * failure. */
} Tcl_ObjType;

```

```

/*
 * One of the following structures exists for each object in the Tcl
 * system. An object stores a value as either a string, some internal
 * representation, or both.
 */

```

```

typedef struct Tcl_Obj {
    int refCount; /* When 0 the object will be freed. */
    char *bytes; /* This points to the first byte of the
     * object's string representation. The array
     * must be followed by a null byte (i.e., at
     * offset length) but may also contain
     * embedded null characters. The array's
     * storage is allocated by ckalloc. NULL
     * means the string rep is invalid and must
     * be regenerated from the internal rep.
     * Clients should use Tcl_GetStringFromObj

```



```

        * to get a pointer to the byte array as a
        * readonly value. */
int length;          /* The number of bytes at *bytes, not
        * including the terminating null. */
Tcl_ObjType *typePtr; /* Denotes the object's type. Always
        * corresponds to the type of the object's
        * internal rep. NULL indicates the object
        * has no internal rep (has no type). */
union {              /* The internal representation: */
    long longValue;   /* - an long integer value */
    double doubleValue; /* - a double-precision floating value */
    VOID *otherValuePtr; /* - another, type-specific value */
    struct {          /* - internal rep as two pointers */
        VOID *ptr1;
        VOID *ptr2;
    } twoPtrValue;
    } internalRep;
} Tcl_Obj;

/*
 * Macros to increment and decrement a Tcl_Obj's reference count, and to
 * test whether an object is shared (i.e. has reference count > 1).
 * Note: clients should use Tcl_DecrRefCount() when they are finished using
 * an object, and should never call TclFreeObj() directly. TclFreeObj() is
 * only defined and made public in tcl.h to support Tcl_DecrRefCount's macro
 * definition. Note also that Tcl_DecrRefCount() refers to the parameter
 * "obj" twice. This means that you should avoid calling it with an
 * expression that is expensive to compute or has side effects.
 */

EXTERN void      Tcl_IncrRefCount _ANSI_ARGS_((Tcl_Obj *objPtr));
EXTERN void      Tcl_DecrRefCount _ANSI_ARGS_((Tcl_Obj *objPtr));
EXTERN int       Tcl_IsShared _ANSI_ARGS_((Tcl_Obj *objPtr));

#ifdef TCL_MEM_DEBUG
#   define Tcl_IncrRefCount(objPtr) \
        Tcl_DbIncrRefCount(objPtr, __FILE__, __LINE__)
#   define Tcl_DecrRefCount(objPtr) \
        Tcl_DbDecrRefCount(objPtr, __FILE__, __LINE__)
#   define Tcl_IsShared(objPtr) \
        Tcl_DbIsShared(objPtr, __FILE__, __LINE__)
#else
#   define Tcl_IncrRefCount(objPtr) \
        ++(objPtr)->refCount
#   define Tcl_DecrRefCount(objPtr) \
        if (--(objPtr)->refCount <= 0) TclFreeObj(objPtr)
#   define Tcl_IsShared(objPtr) \
        ((objPtr)->refCount > 1)
#endif

/*
 * Macros and definitions that help to debug the use of Tcl objects.
 * When TCL_MEM_DEBUG is defined, the Tcl_New* declarations are
 * overridden to call debugging versions of the object creation procedures.
 */

EXTERN Tcl_Obj * Tcl_NewBooleanObj _ANSI_ARGS_((int boolValue));

```

```

EXTERN Tcl_Obj * Tcl_NewDoubleObj _ANSI_ARGS_((double doubleValue));
EXTERN Tcl_Obj * Tcl_NewIntObj _ANSI_ARGS_((int intValue));
EXTERN Tcl_Obj * Tcl_NewListObj _ANSI_ARGS_((int objc,
      Tcl_Obj *CONST objv[]));
EXTERN Tcl_Obj * Tcl_NewLongObj _ANSI_ARGS_((long longValue));
EXTERN Tcl_Obj * Tcl_NewObj _ANSI_ARGS_((void));
EXTERN Tcl_Obj * Tcl_NewStringObj _ANSI_ARGS_((char *bytes,
      int length));

```

```

#ifdef TCL_MEM_DEBUG
# define Tcl_NewBooleanObj(val) \
    Tcl_DbNewBooleanObj(val, __FILE__, __LINE__)
# define Tcl_NewDoubleObj(val) \
    Tcl_DbNewDoubleObj(val, __FILE__, __LINE__)
# define Tcl_NewIntObj(val) \
    Tcl_DbNewLongObj(val, __FILE__, __LINE__)
# define Tcl_NewListObj(objc, objv) \
    Tcl_DbNewListObj(objc, objv, __FILE__, __LINE__)
# define Tcl_NewLongObj(val) \
    Tcl_DbNewLongObj(val, __FILE__, __LINE__)
# define Tcl_NewObj() \
    Tcl_DbNewObj(__FILE__, __LINE__)
# define Tcl_NewStringObj(bytes, len) \
    Tcl_DbNewStringObj(bytes, len, __FILE__, __LINE__)
#endif /* TCL_MEM_DEBUG */

```

```

/*
 * The following definitions support Tcl's namespace facility.
 * Note: the first five fields must match exactly the fields in a
 * Namespace structure (see tcl.h).
 */

```

```

typedef struct Tcl_Namespace {
    char *name; /* The namespace's name within its parent
                * namespace. This contains no ::'s. The
                * name of the global namespace is ""
                * although "::" is an synonym. */
    char *fullName; /* The namespace's fully qualified name.
                    * This starts with ::. */
    ClientData clientData; /* Arbitrary value associated with this
                            * namespace. */
    Tcl_NamespaceDeleteProc* deleteProc; /* Procedure invoked when deleting the
                                           * namespace to, e.g., free clientData. */
    struct Tcl_Namespace* parentPtr; /* Points to the namespace that contains
                                      * this one. NULL if this is the global
                                      * namespace. */
} Tcl_Namespace;

```

```

/*
 * The following structure represents a call frame, or activation record.
 * A call frame defines a naming context for a procedure call: its local
 * scope (for local variables) and its namespace scope (used for non-local
 * variables; often the global :: namespace). A call frame can also define
 * the naming context for a namespace eval or namespace inscope command:
 * the namespace in which the command's code should execute. The

```

```

* Tcl_CallFrame structures exist only while procedures or namespace
* eval/inscope's are being executed, and provide a Tcl call stack.
*
* A call frame is initialized and pushed using Tcl_PushCallFrame and
* popped using Tcl_PopCallFrame. Storage for a Tcl_CallFrame must be
* provided by the Tcl_PushCallFrame caller, and callers typically allocate
* them on the C call stack for efficiency. For this reason, Tcl_CallFrame
* is defined as a structure and not as an opaque token. However, most
* Tcl_CallFrame fields are hidden since applications should not access
* them directly; others are declared as "dummyX".
*
* WARNING!! The structure definition must be kept consistent with the
* CallFrame structure in tclInt.h. If you change one, change the other.
*/

```

```

typedef struct Tcl_CallFrame {
    Tcl_Namespace *nsPtr;
    int dummy1;
    int dummy2;
    char *dummy3;
    char *dummy4;
    char *dummy5;
    int dummy6;
    char *dummy7;
    char *dummy8;
    int dummy9;
    char* dummy10;
} Tcl_CallFrame;

```

```

/*
* Information about commands that is returned by Tcl_GetCommandInfo and
* passed to Tcl_SetCommandInfo. objProc is an objc/objv object-based
* command procedure while proc is a traditional Tcl argc/argv
* string-based procedure. Tcl_CreateObjCommand and Tcl_CreateCommand
* ensure that both objProc and proc are non-NULL and can be called to
* execute the command. However, it may be faster to call one instead of
* the other. The member isNativeObjectProc is set to 1 if an
* object-based procedure was registered by Tcl_CreateObjCommand, and to
* 0 if a string-based procedure was registered by Tcl_CreateCommand.
* The other procedure is typically set to a compatibility wrapper that
* does string-to-object or object-to-string argument conversions then
* calls the other procedure.
*/

```

```

typedef struct Tcl_CmdInfo {
    int isNativeObjectProc; /* 1 if objProc was registered by a call to
                           * Tcl_CreateObjCommand; 0 otherwise.
                           * Tcl_SetCmdInfo does not modify this
                           * field. */
    Tcl_ObjCmdProc *objProc; /* Command's object-based procedure. */
    ClientData objClientData; /* ClientData for object proc. */
    Tcl_CmdProc *proc; /* Command's string-based procedure. */
    ClientData clientData; /* ClientData for string proc. */
    Tcl_CmdDeleteProc *deleteProc;
                           /* Procedure to call when command is
                           * deleted. */
    ClientData deleteData; /* Value to pass to deleteProc (usually

```

```

        * the same as clientData). */
    Tcl_Namespace *namespacePtr; /* Points to the namespace that contains
        * this command. Note that Tcl_SetCmdInfo
        * will not change a command's namespace;
        * use Tcl_RenameCommand to do that. */
} Tcl_CmdInfo;

/*
 * The structure defined below is used to hold dynamic strings. The only
 * field that clients should use is the string field, and they should
 * never modify it.
 */

#define TCL_DSTRING_STATIC_SIZE 200
typedef struct Tcl_DString {
    char *string;          /* Points to beginning of string: either
        * staticSpace below or a malloced array. */
    int length;            /* Number of non-NULL characters in the
        * string. */
    int spaceAvl;          /* Total number of bytes available for the
        * string and its terminating NULL char. */
    char staticSpace[TCL_DSTRING_STATIC_SIZE];
        /* Space to use in common case where string
        * is small. */
} Tcl_DString;

#define Tcl_DStringLength(dsPtr) ((dsPtr)->length)
#define Tcl_DStringValue(dsPtr) ((dsPtr)->string)
#define Tcl_DStringTrunc Tcl_DStringSetLength

/*
 * Definitions for the maximum number of digits of precision that may
 * be specified in the "tcl_precision" variable, and the number of
 * characters of buffer space required by Tcl_PrintDouble.
 */

#define TCL_MAX_PREC 17
#define TCL_DOUBLE_SPACE (TCL_MAX_PREC+10)

/*
 * Flag that may be passed to Tcl_ConvertElement to force it not to
 * output braces (careful! if you change this flag be sure to change
 * the definitions at the front of tclUtil.c).
 */

#define TCL_DONT_USE_BRACES 1

/*
 * Flag that may be passed to Tcl_GetIndexFromObj to force it to disallow
 * abbreviated strings.
 */

#define TCL_EXACT 1

/*
 * Flag values passed to Tcl_RecordAndEval.
 */

```

```

* WARNING: these bit choices must not conflict with the bit choices
* for evalFlag bits in tclInt.h!!
*/

#define TCL_NO_EVAL            0x10000
#define TCL_EVAL_GLOBAL        0x20000

/*
 * Special freeProc values that may be passed to Tcl_SetResult (see
 * the man page for details):
 */

#define TCL_VOLATILE           ((Tcl_FreeProc *) 1)
#define TCL_STATIC             ((Tcl_FreeProc *) 0)
#define TCL_DYNAMIC            ((Tcl_FreeProc *) 3)

/*
 * Flag values passed to variable-related procedures.
 */

#define TCL_GLOBAL_ONLY        1
#define TCL_NAMESPACE_ONLY     2
#define TCL_APPEND_VALUE       4
#define TCL_LIST_ELEMENT       8
#define TCL_TRACE_READS        0x10
#define TCL_TRACE_WRITES       0x20
#define TCL_TRACE_UNSETS       0x40
#define TCL_TRACE_DESTROYED    0x80
#define TCL_INTERP_DESTROYED   0x100
#define TCL_LEAVE_ERR_MSG      0x200
#define TCL_PARSE_PART1        0x400

/*
 * Types for linked variables:
 */

#define TCL_LINK_INT           1
#define TCL_LINK_DOUBLE        2
#define TCL_LINK_BOOLEAN       3
#define TCL_LINK_STRING         4
#define TCL_LINK_READ_ONLY     0x80

/*
 * The following declarations either map ckalloc and ckfree to
 * malloc and free, or they map them to procedures with all sorts
 * of debugging hooks defined in tclCkalloc.c.
 */

EXTERN char *      Tcl_Alloc _ANSI_ARGS_((unsigned int size));
EXTERN void        Tcl_Free _ANSI_ARGS_((char *ptr));
EXTERN char *      Tcl_Realloc _ANSI_ARGS_((char *ptr,
                                           unsigned int size));

#ifdef TCL_MEM_DEBUG

# define Tcl_Alloc(x) Tcl_DbCkalloc(x, __FILE__, __LINE__)
# define Tcl_Free(x)  Tcl_DbCkfree(x, __FILE__, __LINE__)


```

[illegible]

```

ClientData clientData;          /* Application stores something here
                                * with Tcl_SetHashValue. */
union {
    char *oneWordValue;          /* Key has one of these forms: */
    int words[1];                /* One-word value for key. */
                                /* Multiple integer words for key.
                                * The actual size will be as large
                                * as necessary for this table's
                                * keys. */
    char string[4];              /* String for key. The actual size
                                * will be as large as needed to hold
                                * the key. */
} key;                           /* MUST BE LAST FIELD IN RECORD!! */
} Tcl_HashEntry;

/*
 * Structure definition for a hash table. Must be in tcl.h so clients
 * can allocate space for these structures, but clients should never
 * access any fields in this structure.
 */

#define TCL_SMALL_HASH_TABLE 4
typedef struct Tcl_HashTable {
    Tcl_HashEntry **buckets;      /* Pointer to bucket array. Each
                                * element points to first entry in
                                * bucket's hash chain, or NULL. */
    Tcl_HashEntry *staticBuckets[TCL_SMALL_HASH_TABLE];
                                /* Bucket array used for small tables
                                * (to avoid mallocs and frees). */
    int numBuckets;               /* Total number of buckets allocated
                                * at **bucketPtr. */
    int numEntries;               /* Total number of entries present
                                * in table. */
    int rebuildSize;              /* Enlarge table when numEntries gets
                                * to be this large. */
    int downShift;                /* Shift count used in hashing
                                * function. Designed to use high-
                                * order bits of randomized keys. */
    int mask;                     /* Mask value used in hashing
                                * function. */
    int keyType;                  /* Type of keys used in this table.
                                * It's either TCL_STRING_KEYS,
                                * TCL_ONE_WORD_KEYS, or an integer
                                * giving the number of ints that
                                * is the size of the key.
                                */
    Tcl_HashEntry *(*findProc) _ANSI_ARGS_((struct Tcl_HashTable *tablePtr,
                                CONST char *key));
    Tcl_HashEntry *(*createProc) _ANSI_ARGS_((struct Tcl_HashTable *tablePtr,
                                CONST char *key, int *newPtr));
} Tcl_HashTable;

/*
 * Structure definition for information used to keep track of searches
 * through hash tables:
 */

typedef struct Tcl_HashSearch {

```

```

    Tcl_HashTable *tablePtr;          /* Table being searched. */
    int nextIndex;                    /* Index of next bucket to be
                                     * enumerated after present one. */
    Tcl_HashEntry *nextEntryPtr;      /* Next entry to be enumerated in the
                                     * the current bucket. */
} Tcl_HashSearch;

/*
 * Acceptable key types for hash tables:
 */

#define TCL_STRING_KEYS      0
#define TCL_ONE_WORD_KEYS   1

/*
 * Macros for clients to use to access fields of hash entries:
 */

#define Tcl_GetHashValue(h) ((h)->clientData)
#define Tcl_SetHashValue(h, value) ((h)->clientData = (ClientData) (value))
#define Tcl_GetHashKey(tablePtr, h) \
    ((char *) (((tablePtr)->keyType == TCL_ONE_WORD_KEYS) ? (h)->key.oneWordValue \
                                                         : (h)->key.string))

/*
 * Macros to use for clients to use to invoke find and create procedures
 * for hash tables:
 */

#define Tcl_FindHashEntry(tablePtr, key) \
    (*((tablePtr)->findProc))(tablePtr, key)
#define Tcl_CreateHashEntry(tablePtr, key, newPtr) \
    (*((tablePtr)->createProc))(tablePtr, key, newPtr)

/*
 * Flag values to pass to Tcl_DoOneEvent to disable searches
 * for some kinds of events:
 */

#define TCL_DONT_WAIT      (1<<1)
#define TCL_WINDOW_EVENTS (1<<2)
#define TCL_FILE_EVENTS   (1<<3)
#define TCL_TIMER_EVENTS  (1<<4)
#define TCL_IDLE_EVENTS   (1<<5) /* WAS 0x10 ???? */
#define TCL_ALL_EVENTS    (~TCL_DONT_WAIT)

/*
 * The following structure defines a generic event for the Tcl event
 * system. These are the things that are queued in calls to Tcl_QueueEvent
 * and serviced later by Tcl_DoOneEvent. There can be many different
 * kinds of events with different fields, corresponding to window events,
 * timer events, etc. The structure for a particular event consists of
 * a Tcl_Event header followed by additional information specific to that
 * event.
 */

```



```

struct Tcl_Event {
    Tcl_EventProc *proc;      /* Procedure to call to service this event. */
    struct Tcl_Event *nextPtr; /* Next in list of pending events, or NULL. */
};

/*
 * Positions to pass to Tcl_QueueEvent:
 */

typedef enum {
    TCL_QUEUE_TAIL, TCL_QUEUE_HEAD, TCL_QUEUE_MARK
} Tcl_QueuePosition;

/*
 * Values to pass to Tcl_SetServiceMode to specify the behavior of notifier
 * event routines.
 */

#define TCL_SERVICE_NONE 0
#define TCL_SERVICE_ALL 1

/*
 * The following structure keeps is used to hold a time value, either as
 * an absolute time (the number of seconds from the epoch) or as an
 * elapsed time. On Unix systems the epoch is Midnight Jan 1, 1970 GMT.
 * On Macintosh systems the epoch is Midnight Jan 1, 1904 GMT.
 */

typedef struct Tcl_Time {
    long sec;          /* Seconds. */
    long usec;         /* Microseconds. */
} Tcl_Time;

/*
 * Bits to pass to Tcl_CreateFileHandler and Tcl_CreateChannelHandler
 * to indicate what sorts of events are of interest:
 */

#define TCL_READABLE      (1<<1)
#define TCL_WRITABLE      (1<<2)
#define TCL_EXCEPTION      (1<<3)

/*
 * Flag values to pass to Tcl_OpenCommandChannel to indicate the
 * disposition of the stdio handles.  TCL_STDIN, TCL_STDOUT, TCL_STDERR,
 * are also used in Tcl_GetStdChannel.
 */

#define TCL_STDIN          (1<<1)
#define TCL_STDOUT          (1<<2)
#define TCL_STDERR          (1<<3)
#define TCL_ENFORCE_MODE    (1<<4)

/*
 * Typedefs for the various operations in a channel type:
 */

```

[illegible]

```

    Tcl_DriverSetOptionProc *setOptionProc;
                                /* Set an option on a channel. */
    Tcl_DriverGetOptionProc *getOptionProc;
                                /* Get an option from a channel. */
    Tcl_DriverWatchProc *watchProc; /* Set up the notifier to watch
                                    * for events on this channel. */
    Tcl_DriverGetHandleProc *getHandleProc;
                                /* Get an OS handle from the channel
                                    * or NULL if not supported. */
    VOID *reserved;             /* reserved for future expansion */
} Tcl_ChannelType;

/*
 * The following flags determine whether the blockModeProc above should
 * set the channel into blocking or nonblocking mode. They are passed
 * as arguments to the blockModeProc procedure in the above structure.
 */

#define TCL_MODE_BLOCKING 0      /* Put channel into blocking mode. */
#define TCL_MODE_NONBLOCKING 1  /* Put channel into nonblocking
                                * mode. */

/*
 * Enum for different types of file paths.
 */

typedef enum Tcl_PathType {
    TCL_PATH_ABSOLUTE,
    TCL_PATH_RELATIVE,
    TCL_PATH_VOLUME_RELATIVE
} Tcl_PathType;

/*
 * Exported Tcl procedures:
 */

EXTERN void      Tcl_AddErrorInfo _ANSI_ARGS_((Tcl_Interp *interp,
        char *message));
EXTERN void      Tcl_AddObjErrorInfo _ANSI_ARGS_((Tcl_Interp *interp,
        char *message, int length));
EXTERN void      Tcl-AllowExceptions _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN int       Tcl_AppendAllObjTypes _ANSI_ARGS_((
        Tcl_Interp *interp, Tcl_Obj *objPtr));
EXTERN void      Tcl_AppendElement _ANSI_ARGS_((Tcl_Interp *interp,
        char *string));
EXTERN void      Tcl_AppendResult _ANSI_ARGS_((
        TCL_VARARGS(Tcl_Interp *,interp));
EXTERN void      Tcl_AppendToObj _ANSI_ARGS_((Tcl_Obj *objPtr,
        char *bytes, int length));
EXTERN void      Tcl_AppendStringsToObj _ANSI_ARGS_((
        TCL_VARARGS(Tcl_Obj *,interp));
EXTERN Tcl_AsyncHandler Tcl_AsyncCreate _ANSI_ARGS_((Tcl_AsyncProc *proc,
        ClientData clientData));
EXTERN void      Tcl_AsyncDelete _ANSI_ARGS_((Tcl_AsyncHandler async));
EXTERN int       Tcl_AsyncInvoke _ANSI_ARGS_((Tcl_Interp *interp,
        int code));
EXTERN void      Tcl_AsyncMark _ANSI_ARGS_((Tcl_AsyncHandler async));

```

```

EXTERN int      Tcl_AsyncReady _ANSI_ARGS_((void));
EXTERN void     Tcl_BackgroundError _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN char     Tcl_Backslash _ANSI_ARGS_((CONST char *src,
      int *readPtr));
EXTERN int      Tcl_BadChannelOption _ANSI_ARGS_((Tcl_Interp *interp,
      char *optionName, char *optionList));
EXTERN void     Tcl_CallWhenDeleted _ANSI_ARGS_((Tcl_Interp *interp,
      Tcl_InterpDeleteProc *proc,
      ClientData clientData));
EXTERN void     Tcl_CancelIdleCall _ANSI_ARGS_((Tcl_IdleProc *idleProc,
      ClientData clientData));
#define Tcl_Ckalloc Tcl_Alloc
#define Tcl_Ckfree Tcl_Free
#define Tcl_Ckrealloc Tcl_Realloc
EXTERN int      Tcl_Close _ANSI_ARGS_((Tcl_Interp *interp,
      Tcl_Channel chan));
EXTERN int      Tcl_CommandComplete _ANSI_ARGS_((char *cmd));
EXTERN char *   Tcl_Concat _ANSI_ARGS_((int argc, char **argv));
EXTERN Tcl_Obj * Tcl_ConcatObj _ANSI_ARGS_((int objc,
      Tcl_Obj *CONST objv[]));
EXTERN int      Tcl_ConvertCountedElement _ANSI_ARGS_((CONST char *src,
      int length, char *dst, int flags));
EXTERN int      Tcl_ConvertElement _ANSI_ARGS_((CONST char *src,
      char *dst, int flags));
EXTERN int      Tcl_ConvertToType _ANSI_ARGS_((Tcl_Interp *interp,
      Tcl_Obj *objPtr, Tcl_ObjType *typePtr));
EXTERN int      Tcl_CreateAlias _ANSI_ARGS_((Tcl_Interp *slave,
      char *slaveCmd, Tcl_Interp *target,
      char *targetCmd, int argc, char **argv));
EXTERN int      Tcl_CreateAliasObj _ANSI_ARGS_((Tcl_Interp *slave,
      char *slaveCmd, Tcl_Interp *target,
      char *targetCmd, int objc,
      Tcl_Obj *CONST objv[]));
EXTERN Tcl_Channel Tcl_CreateChannel _ANSI_ARGS_((
      Tcl_ChannelType *typePtr, char *chanName,
      ClientData instanceData, int mask));
EXTERN void     Tcl_CreateChannelHandler _ANSI_ARGS_((
      Tcl_Channel chan, int mask,
      Tcl_ChannelProc *proc, ClientData clientData));
EXTERN void     Tcl_CreateCloseHandler _ANSI_ARGS_((
      Tcl_Channel chan, Tcl_CloseProc *proc,
      ClientData clientData));
EXTERN Tcl_Command Tcl_CreateCommand _ANSI_ARGS_((Tcl_Interp *interp,
      char *cmdName, Tcl_CmdProc *proc,
      ClientData clientData,
      Tcl_CmdDeleteProc *deleteProc));
EXTERN void     Tcl_CreateEventSource _ANSI_ARGS_((
      Tcl_EventSetupProc *setupProc,
      Tcl_EventCheckProc *checkProc,
      ClientData clientData));
EXTERN void     Tcl_CreateExitHandler _ANSI_ARGS_((Tcl_ExitProc *proc,
      ClientData clientData));
EXTERN void     Tcl_CreateFileHandler _ANSI_ARGS_((
      int fd, int mask, Tcl_FileProc *proc,
      ClientData clientData));
EXTERN Tcl_Interp * Tcl_CreateInterp _ANSI_ARGS_((void));
EXTERN void     Tcl_CreateMathFunc _ANSI_ARGS_((Tcl_Interp *interp,

```

```

char *name, int numArgs, Tcl_ValueType *argTypes,
Tcl_MathProc *proc, ClientData clientData));
EXTERN Tcl_Command Tcl_CreateObjCommand _ANSI_ARGS_((
Tcl_Interp *interp, char *cmdName,
Tcl_ObjCmdProc *proc, ClientData clientData,
Tcl_CmdDeleteProc *deleteProc));
EXTERN Tcl_Interp * Tcl_CreateSlave _ANSI_ARGS_((Tcl_Interp *interp,
char *slaveName, int isSafe));
EXTERN Tcl_TimerToken Tcl_CreateTimerHandler _ANSI_ARGS_((int milliseconds,
Tcl_TimerProc *proc, ClientData clientData));
EXTERN Tcl_Trace Tcl_CreateTrace _ANSI_ARGS_((Tcl_Interp *interp,
int level, Tcl_CmdTraceProc *proc,
ClientData clientData));
EXTERN char * Tcl_DbCkalloc _ANSI_ARGS_((unsigned int size,
char *file, int line));
EXTERN int Tcl_DbCkfree _ANSI_ARGS_((char *ptr,
char *file, int line));
EXTERN char * Tcl_DbCkrealloc _ANSI_ARGS_((char *ptr,
unsigned int size, char *file, int line));
EXTERN void Tcl_DbDecrRefCount _ANSI_ARGS_((Tcl_Obj *objPtr,
char *file, int line));
EXTERN void Tcl_DbIncrRefCount _ANSI_ARGS_((Tcl_Obj *objPtr,
char *file, int line));
EXTERN int Tcl_DbIsShared _ANSI_ARGS_((Tcl_Obj *objPtr,
char *file, int line));
EXTERN Tcl_Obj * Tcl_DbNewBooleanObj _ANSI_ARGS_((int boolValue,
char *file, int line));
EXTERN Tcl_Obj * Tcl_DbNewDoubleObj _ANSI_ARGS_((double doubleValue,
char *file, int line));
EXTERN Tcl_Obj * Tcl_DbNewListObj _ANSI_ARGS_((int objc,
Tcl_Obj *CONST objv[], char *file, int line));
EXTERN Tcl_Obj * Tcl_DbNewLongObj _ANSI_ARGS_((long longValue,
char *file, int line));
EXTERN Tcl_Obj * Tcl_DbNewObj _ANSI_ARGS_((char *file, int line));
EXTERN Tcl_Obj * Tcl_DbNewStringObj _ANSI_ARGS_((char *bytes,
int length, char *file, int line));
EXTERN void Tcl_DeleteAssocData _ANSI_ARGS_((Tcl_Interp *interp,
char *name));
EXTERN int Tcl_DeleteCommand _ANSI_ARGS_((Tcl_Interp *interp,
char *cmdName));
EXTERN int Tcl_DeleteCommandFromToken _ANSI_ARGS_((
Tcl_Interp *interp, Tcl_Command command));
EXTERN void Tcl_DeleteChannelHandler _ANSI_ARGS_((
Tcl_Channel chan, Tcl_ChannelProc *proc,
ClientData clientData));
EXTERN void Tcl_DeleteCloseHandler _ANSI_ARGS_((
Tcl_Channel chan, Tcl_CloseProc *proc,
ClientData clientData));
EXTERN void Tcl_DeleteEvents _ANSI_ARGS_((
Tcl_EventDeleteProc *proc,
ClientData clientData));
EXTERN void Tcl_DeleteEventSource _ANSI_ARGS_((
Tcl_EventSetupProc *setupProc,
Tcl_EventCheckProc *checkProc,
ClientData clientData));
EXTERN void Tcl_DeleteExitHandler _ANSI_ARGS_((Tcl_ExitProc *proc,
ClientData clientData));

```

```

EXTERN void      Tcl_DeleteFileHandler _ANSI_ARGS_((int fd));
EXTERN void      Tcl_DeleteHashEntry _ANSI_ARGS_((
    Tcl_HashEntry *entryPtr));
EXTERN void      Tcl_DeleteHashTable _ANSI_ARGS_((
    Tcl_HashTable *tablePtr));
EXTERN void      Tcl_DeleteInterp _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN void      Tcl_DeleteTimerHandler _ANSI_ARGS_((
    Tcl_TimerToken token));
EXTERN void      Tcl_DeleteTrace _ANSI_ARGS_((Tcl_Interp *interp,
    Tcl_Trace trace));
EXTERN void      Tcl_DetachPids _ANSI_ARGS_((int numPids, Tcl_Pid *pidPtr));
EXTERN void      Tcl_DontCallWhenDeleted _ANSI_ARGS_((
    Tcl_Interp *interp, Tcl_InterpDeleteProc *proc,
    ClientData clientData));
EXTERN int       Tcl_DoOneEvent _ANSI_ARGS_((int flags));
EXTERN void      Tcl_DoWhenIdle _ANSI_ARGS_((Tcl_IdleProc *proc,
    ClientData clientData));
EXTERN char *    Tcl_DStringAppend _ANSI_ARGS_((Tcl_DString *dsPtr,
    CONST char *string, int length));
EXTERN char *    Tcl_DStringAppendElement _ANSI_ARGS_((
    Tcl_DString *dsPtr, CONST char *string));
EXTERN void      Tcl_DStringEndSublist _ANSI_ARGS_((Tcl_DString *dsPtr));
EXTERN void      Tcl_DStringFree _ANSI_ARGS_((Tcl_DString *dsPtr));
EXTERN void      Tcl_DStringGetResult _ANSI_ARGS_((Tcl_Interp *interp,
    Tcl_DString *dsPtr));
EXTERN void      Tcl_DStringInit _ANSI_ARGS_((Tcl_DString *dsPtr));
EXTERN void      Tcl_DStringResult _ANSI_ARGS_((Tcl_Interp *interp,
    Tcl_DString *dsPtr));
EXTERN void      Tcl_DStringSetLength _ANSI_ARGS_((Tcl_DString *dsPtr,
    int length));
EXTERN void      Tcl_DStringStartSublist _ANSI_ARGS_((
    Tcl_DString *dsPtr));
EXTERN Tcl_Obj * Tcl_DuplicateObj _ANSI_ARGS_((Tcl_Obj *objPtr));
EXTERN int       Tcl_Eof _ANSI_ARGS_((Tcl_Channel chan));
EXTERN char *    Tcl_ErrnoId _ANSI_ARGS_((void));
EXTERN char *    Tcl_ErrnoMsg _ANSI_ARGS_((int err));
EXTERN int       Tcl_Eval _ANSI_ARGS_((Tcl_Interp *interp,
    char *string));
EXTERN int       Tcl_EvalFile _ANSI_ARGS_((Tcl_Interp *interp,
    char *fileName));
EXTERN void      Tcl_EventuallyFree _ANSI_ARGS_((ClientData clientData,
    Tcl_FreeProc *freeProc));
EXTERN int       Tcl_EvalObj _ANSI_ARGS_((Tcl_Interp *interp,
    Tcl_Obj *objPtr));
EXTERN void      Tcl_Exit _ANSI_ARGS_((int status));
EXTERN int       Tcl_ExposeCommand _ANSI_ARGS_((Tcl_Interp *interp,
    char *hiddenCmdToken, char *cmdName));
EXTERN int       Tcl_ExprBoolean _ANSI_ARGS_((Tcl_Interp *interp,
    char *string, int *ptr));
EXTERN int       Tcl_ExprBooleanObj _ANSI_ARGS_((Tcl_Interp *interp,
    Tcl_Obj *objPtr, int *ptr));
EXTERN int       Tcl_ExprDouble _ANSI_ARGS_((Tcl_Interp *interp,
    char *string, double *ptr));
EXTERN int       Tcl_ExprDoubleObj _ANSI_ARGS_((Tcl_Interp *interp,
    Tcl_Obj *objPtr, double *ptr));
EXTERN int       Tcl_ExprLong _ANSI_ARGS_((Tcl_Interp *interp,
    char *string, long *ptr));

```

```

EXTERN int      Tcl_ExprLongObj _ANSI_ARGS_((Tcl_Interp *interp,
      Tcl_Obj *objPtr, long *ptr));
EXTERN int      Tcl_ExprObj _ANSI_ARGS_((Tcl_Interp *interp,
      Tcl_Obj *objPtr, Tcl_Obj **resultPtrPtr));
EXTERN int      Tcl_ExprString _ANSI_ARGS_((Tcl_Interp *interp,
      char *string));
EXTERN void     Tcl_Finalize _ANSI_ARGS_((void));
EXTERN void     Tcl_FindExecutable _ANSI_ARGS_((char *argv0));
EXTERN Tcl_HashEntry * Tcl_FirstHashEntry _ANSI_ARGS_((
      Tcl_HashTable *tablePtr,
      Tcl_HashSearch *searchPtr));
EXTERN int      Tcl_Flush _ANSI_ARGS_((Tcl_Channel chan));
EXTERN void     TclFreeObj _ANSI_ARGS_((Tcl_Obj *objPtr));
EXTERN void     Tcl_FreeResult _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN int      Tcl_GetAlias _ANSI_ARGS_((Tcl_Interp *interp,
      char *slaveCmd, Tcl_Interp **targetInterpPtr,
      char **targetCmdPtr, int *argcPtr,
      char ***argvPtr));
EXTERN int      Tcl_GetAliasObj _ANSI_ARGS_((Tcl_Interp *interp,
      char *slaveCmd, Tcl_Interp **targetInterpPtr,
      char **targetCmdPtr, int *objcPtr,
      Tcl_Obj ***objv));
EXTERN ClientData Tcl_GetAssocData _ANSI_ARGS_((Tcl_Interp *interp,
      char *name, Tcl_InterpDeleteProc **procPtr));
EXTERN int      Tcl_GetBoolean _ANSI_ARGS_((Tcl_Interp *interp,
      char *string, int *boolPtr));
EXTERN int      Tcl_GetBooleanFromObj _ANSI_ARGS_((
      Tcl_Interp *interp, Tcl_Obj *objPtr,
      int *boolPtr));
EXTERN Tcl_Channel Tcl_GetChannel _ANSI_ARGS_((Tcl_Interp *interp,
      char *chanName, int *modePtr));
EXTERN int      Tcl_GetChannelBufferSize _ANSI_ARGS_((
      Tcl_Channel chan));
EXTERN int      Tcl_GetChannelHandle _ANSI_ARGS_((Tcl_Channel chan,
      int direction, ClientData *handlePtr));
EXTERN ClientData Tcl_GetChannelInstanceData _ANSI_ARGS_((
      Tcl_Channel chan));
EXTERN int      Tcl_GetChannelMode _ANSI_ARGS_((Tcl_Channel chan));
EXTERN char *   Tcl_GetChannelName _ANSI_ARGS_((Tcl_Channel chan));
EXTERN int      Tcl_GetChannelOption _ANSI_ARGS_((Tcl_Interp *interp,
      Tcl_Channel chan, char *optionName,
      Tcl_DString *dsPtr));
EXTERN Tcl_ChannelType * Tcl_GetChannelType _ANSI_ARGS_((Tcl_Channel chan));
EXTERN int      Tcl_GetCommandInfo _ANSI_ARGS_((Tcl_Interp *interp,
      char *cmdName, Tcl_CmdInfo *infoPtr));
EXTERN char *   Tcl_GetCommandName _ANSI_ARGS_((Tcl_Interp *interp,
      Tcl_Command command));
EXTERN int      Tcl_GetDouble _ANSI_ARGS_((Tcl_Interp *interp,
      char *string, double *doublePtr));
EXTERN int      Tcl_GetDoubleFromObj _ANSI_ARGS_((
      Tcl_Interp *interp, Tcl_Obj *objPtr,
      double *doublePtr));
EXTERN int      Tcl_GetErrno _ANSI_ARGS_((void));
EXTERN char *   Tcl_GetHostName _ANSI_ARGS_((void));
EXTERN int      Tcl_GetIndexFromObj _ANSI_ARGS_((Tcl_Interp *interp,
      Tcl_Obj *objPtr, char **tablePtr, char *msg,
      int flags, int *indexPtr));

```

```

EXTERN int      Tcl_GetInt _ANSI_ARGS_((Tcl_Interp *interp,
char *string, int *intPtr));
EXTERN int      Tcl_GetInterpPath _ANSI_ARGS_((Tcl_Interp *askInterp,
Tcl_Interp *slaveInterp));
EXTERN int      Tcl_GetIntFromObj _ANSI_ARGS_((Tcl_Interp *interp,
Tcl_Obj *objPtr, int *intPtr));
EXTERN int      Tcl_GetLongFromObj _ANSI_ARGS_((Tcl_Interp *interp,
Tcl_Obj *objPtr, long *longPtr));
EXTERN Tcl_Interp *      Tcl_GetMaster _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN CONST char *      Tcl_GetNameOfExecutable _ANSI_ARGS_((void));
EXTERN Tcl_Obj *      Tcl_GetObjResult _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN Tcl_ObjType *      Tcl_GetObjType _ANSI_ARGS_((char *typeName));
EXTERN int      Tcl_GetOpenFile _ANSI_ARGS_((Tcl_Interp *interp,
char *string, int write, int checkUsage,
ClientData *filePtr));
EXTERN Tcl_PathType      Tcl_GetPathType _ANSI_ARGS_((char *path));
EXTERN int      Tcl_Gets _ANSI_ARGS_((Tcl_Channel chan,
Tcl_DString *dsPtr));
EXTERN int      Tcl_GetsObj _ANSI_ARGS_((Tcl_Channel chan,
Tcl_Obj *objPtr));
EXTERN int      Tcl_GetServiceMode _ANSI_ARGS_((void));
EXTERN Tcl_Interp *      Tcl_GetSlave _ANSI_ARGS_((Tcl_Interp *interp,
char *slaveName));
EXTERN Tcl_Channel      Tcl_GetStdChannel _ANSI_ARGS_((int type));
EXTERN char *      Tcl_GetStringFromObj _ANSI_ARGS_((Tcl_Obj *objPtr,
int *lengthPtr));
EXTERN char *      Tcl_GetStringResult _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN char *      Tcl_GetVar _ANSI_ARGS_((Tcl_Interp *interp,
char *varName, int flags));
EXTERN char *      Tcl_GetVar2 _ANSI_ARGS_((Tcl_Interp *interp,
char *part1, char *part2, int flags));
EXTERN int      Tcl_GlobalEval _ANSI_ARGS_((Tcl_Interp *interp,
char *command));
EXTERN int      Tcl_GlobalEvalObj _ANSI_ARGS_((Tcl_Interp *interp,
Tcl_Obj *objPtr));
EXTERN char *      Tcl_HashStats _ANSI_ARGS_((Tcl_HashTable *tablePtr));
EXTERN int      Tcl_HideCommand _ANSI_ARGS_((Tcl_Interp *interp,
char *cmdName, char *hiddenCmdToken));
EXTERN int      Tcl_Init _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN void      Tcl_InitHashTable _ANSI_ARGS_((Tcl_HashTable *tablePtr,
int keyType));
EXTERN void      Tcl_InitMemory _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN int      Tcl_InputBlocked _ANSI_ARGS_((Tcl_Channel chan));
EXTERN int      Tcl_InputBuffered _ANSI_ARGS_((Tcl_Channel chan));
EXTERN int      Tcl_InterpDeleted _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN int      Tcl_IsSafe _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN void      Tcl_InvalidateStringRep _ANSI_ARGS_((
Tcl_Obj *objPtr));
EXTERN char *      Tcl_JoinPath _ANSI_ARGS_((int argc, char **argv,
Tcl_DString *resultPtr));
EXTERN int      Tcl_LinkVar _ANSI_ARGS_((Tcl_Interp *interp,
char *varName, char *addr, int type));
EXTERN int      Tcl_ListObjAppendList _ANSI_ARGS_((
Tcl_Interp *interp, Tcl_Obj *listPtr,
Tcl_Obj *elemListPtr));
EXTERN int      Tcl_ListObjAppendElement _ANSI_ARGS_((
Tcl_Interp *interp, Tcl_Obj *listPtr,

```



```

        Tcl_Obj *objPtr));
EXTERN int      Tcl_ListObjGetElements _ANSI_ARGS_((
        Tcl_Interp *interp, Tcl_Obj *listPtr,
        int *objcPtr, Tcl_Obj **objvPtr));
EXTERN int      Tcl_ListObjIndex _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_Obj *listPtr, int index,
        Tcl_Obj **objPtrPtr));
EXTERN int      Tcl_ListObjLength _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_Obj *listPtr, int *intPtr));
EXTERN int      Tcl_ListObjReplace _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_Obj *listPtr, int first, int count,
        int objc, Tcl_Obj *CONST objv[]));
EXTERN void     Tcl_Main _ANSI_ARGS_((int argc, char **argv,
        Tcl_AppInitProc *appInitProc));
EXTERN Tcl_Channel Tcl_MakeFileChannel _ANSI_ARGS_((ClientData handle,
        int mode));
EXTERN int      Tcl_MakeSafe _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN Tcl_Channel Tcl_MakeTcpClientChannel _ANSI_ARGS_((
        ClientData tcpSocket));
EXTERN char *   Tcl_Merge _ANSI_ARGS_((int argc, char **argv));
EXTERN Tcl_HashEntry * Tcl_NextHashEntry _ANSI_ARGS_((
        Tcl_HashSearch *searchPtr));
EXTERN void     Tcl_NotifyChannel _ANSI_ARGS_((Tcl_Channel channel,
        int mask));
EXTERN Tcl_Obj * Tcl_ObjGetVar2 _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_Obj *part1Ptr, Tcl_Obj *part2Ptr,
        int flags));
EXTERN Tcl_Obj * Tcl_ObjSetVar2 _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_Obj *part1Ptr, Tcl_Obj *part2Ptr,
        Tcl_Obj *newValuePtr, int flags));
EXTERN Tcl_Channel Tcl_OpenCommandChannel _ANSI_ARGS_((
        Tcl_Interp *interp, int argc, char **argv,
        int flags));
EXTERN Tcl_Channel Tcl_OpenFileChannel _ANSI_ARGS_((Tcl_Interp *interp,
        char *fileName, char *modeString,
        int permissions));
EXTERN Tcl_Channel Tcl_OpenTcpClient _ANSI_ARGS_((Tcl_Interp *interp,
        int port, char *address, char *myaddr,
        int myport, int async));
EXTERN Tcl_Channel Tcl_OpenTcpServer _ANSI_ARGS_((Tcl_Interp *interp,
        int port, char *host,
        Tcl_TcpAcceptProc *acceptProc,
        ClientData callbackData));
EXTERN char *   Tcl_ParseVar _ANSI_ARGS_((Tcl_Interp *interp,
        char *string, char **termPtr));
EXTERN int      Tcl_PkgProvide _ANSI_ARGS_((Tcl_Interp *interp,
        char *name, char *version));
EXTERN char *   Tcl_PkgRequire _ANSI_ARGS_((Tcl_Interp *interp,
        char *name, char *version, int exact));
EXTERN char *   Tcl_PosixError _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN void     Tcl_Preserve _ANSI_ARGS_((ClientData data));
EXTERN void     Tcl_PrintDouble _ANSI_ARGS_((Tcl_Interp *interp,
        double value, char *dst));
EXTERN int      Tcl_PutEnv _ANSI_ARGS_((CONST char *string));
EXTERN void     Tcl_QueueEvent _ANSI_ARGS_((Tcl_Event *evPtr,
        Tcl_QueuePosition position));
EXTERN int      Tcl_Read _ANSI_ARGS_((Tcl_Channel chan,

```

```

        char *bufPtr, int toRead));
EXTERN void      Tcl_ReapDetachedProcs _ANSI_ARGS_((void));
EXTERN int       Tcl_RecordAndEval _ANSI_ARGS_((Tcl_Interp *interp,
        char *cmd, int flags));
EXTERN int       Tcl_RecordAndEvalObj _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_Obj *cmdPtr, int flags));
EXTERN Tcl_RegExp Tcl_RegExpCompile _ANSI_ARGS_((Tcl_Interp *interp,
        char *string));
EXTERN int       Tcl_RegExpExec _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_RegExp regexp, char *string, char *start));
EXTERN int       Tcl_RegExpMatch _ANSI_ARGS_((Tcl_Interp *interp,
        char *string, char *pattern));
EXTERN void      Tcl_RegExpRange _ANSI_ARGS_((Tcl_RegExp regexp,
        int index, char **startPtr, char **endPtr));
EXTERN void      Tcl_RegisterChannel _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_Channel chan));
EXTERN void      Tcl_RegisterObjType _ANSI_ARGS_((
        Tcl_ObjType *typePtr));
EXTERN void      Tcl_Release _ANSI_ARGS_((ClientData clientData));
EXTERN void      Tcl_ResetResult _ANSI_ARGS_((Tcl_Interp *interp));
#define Tcl_Return Tcl_SetResult
EXTERN int       Tcl_ScanCountedElement _ANSI_ARGS_((CONST char *string,
        int length, int *flagPtr));
EXTERN int       Tcl_ScanElement _ANSI_ARGS_((CONST char *string,
        int *flagPtr));
EXTERN int       Tcl_Seek _ANSI_ARGS_((Tcl_Channel chan,
        int offset, int mode));
EXTERN int       Tcl_ServiceAll _ANSI_ARGS_((void));
EXTERN int       Tcl_ServiceEvent _ANSI_ARGS_((int flags));
EXTERN void      Tcl_SetAssocData _ANSI_ARGS_((Tcl_Interp *interp,
        char *name, Tcl_InterpDeleteProc *proc,
        ClientData clientData));
EXTERN void      Tcl_SetBooleanObj _ANSI_ARGS_((Tcl_Obj *objPtr,
        int boolValue));
EXTERN void      Tcl_SetChannelBufferSize _ANSI_ARGS_((
        Tcl_Channel chan, int sz));
EXTERN int       Tcl_SetChannelOption _ANSI_ARGS_((
        Tcl_Interp *interp, Tcl_Channel chan,
        char *optionName, char *newValue));
EXTERN int       Tcl_SetCommandInfo _ANSI_ARGS_((Tcl_Interp *interp,
        char *cmdName, Tcl_CmdInfo *infoPtr));
EXTERN void      Tcl_SetDoubleObj _ANSI_ARGS_((Tcl_Obj *objPtr,
        double doubleValue));
EXTERN void      Tcl_SetErrno _ANSI_ARGS_((int err));
EXTERN void      Tcl_SetErrorCode _ANSI_ARGS_((
        TCL_VARARGS(Tcl_Interp *,arg1));
EXTERN void      Tcl_SetIntObj _ANSI_ARGS_((Tcl_Obj *objPtr,
        int intValue));
EXTERN void      Tcl_SetListObj _ANSI_ARGS_((Tcl_Obj *objPtr,
        int objc, Tcl_Obj *CONST objv[]));
EXTERN void      Tcl_SetLongObj _ANSI_ARGS_((Tcl_Obj *objPtr,
        long longValue));
EXTERN void      Tcl_SetMaxBlockTime _ANSI_ARGS_((Tcl_Time *timePtr));
EXTERN void      Tcl_SetObjErrorCode _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_Obj *errorObjPtr));
EXTERN void      Tcl_SetObjLength _ANSI_ARGS_((Tcl_Obj *objPtr,
        int length));

```

```

EXTERN void      Tcl_SetObjResult _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_Obj *resultObjPtr));
EXTERN void      Tcl_SetPanicProc _ANSI_ARGS_((void (*proc)
        _ANSI_ARGS_(TCL_VARARGS(char *, format))));
EXTERN int       Tcl_SetRecursionLimit _ANSI_ARGS_((Tcl_Interp *interp,
        int depth));
EXTERN void      Tcl_SetResult _ANSI_ARGS_((Tcl_Interp *interp,
        char *string, Tcl_FreeProc *freeProc));
EXTERN int       Tcl_SetServiceMode _ANSI_ARGS_((int mode));
EXTERN void      Tcl_SetStdChannel _ANSI_ARGS_((Tcl_Channel channel,
        int type));
EXTERN void      Tcl_SetStringObj _ANSI_ARGS_((Tcl_Obj *objPtr,
        char *bytes, int length));
EXTERN void      Tcl_SetTimer _ANSI_ARGS_((Tcl_Time *timePtr));
EXTERN char *    Tcl_SetVar _ANSI_ARGS_((Tcl_Interp *interp,
        char *varName, char *newValue, int flags));
EXTERN char *    Tcl_SetVar2 _ANSI_ARGS_((Tcl_Interp *interp,
        char *part1, char *part2, char *newValue,
        int flags));
EXTERN char *    Tcl_SignalId _ANSI_ARGS_((int sig));
EXTERN char *    Tcl_SignalMsg _ANSI_ARGS_((int sig));
EXTERN void      Tcl_Sleep _ANSI_ARGS_((int ms));
EXTERN void      Tcl_SourceRCFile _ANSI_ARGS_((Tcl_Interp *interp));
EXTERN int       Tcl_SplitList _ANSI_ARGS_((Tcl_Interp *interp,
        char *list, int *argcPtr, char ***argvPtr));
EXTERN void      Tcl_SplitPath _ANSI_ARGS_((char *path,
        int *argcPtr, char ***argvPtr));
EXTERN void      Tcl_StaticPackage _ANSI_ARGS_((Tcl_Interp *interp,
        char *pkgName, Tcl_PackageInitProc *initProc,
        Tcl_PackageInitProc *safeInitProc));
EXTERN int       Tcl_StringMatch _ANSI_ARGS_((char *string,
        char *pattern));
EXTERN int       Tcl_Tell _ANSI_ARGS_((Tcl_Channel chan));
#define Tcl_TildeSubst Tcl_TranslateFileName
EXTERN int       Tcl_TraceVar _ANSI_ARGS_((Tcl_Interp *interp,
        char *varName, int flags, Tcl_VarTraceProc *proc,
        ClientData clientData));
EXTERN int       Tcl_TraceVar2 _ANSI_ARGS_((Tcl_Interp *interp,
        char *part1, char *part2, int flags,
        Tcl_VarTraceProc *proc, ClientData clientData));
EXTERN char *    Tcl_TranslateFileName _ANSI_ARGS_((Tcl_Interp *interp,
        char *name, Tcl_DString *bufferPtr));
EXTERN int       Tcl_Ungets _ANSI_ARGS_((Tcl_Channel chan, char *str,
        int len, int atHead));
EXTERN void      Tcl_UnlinkVar _ANSI_ARGS_((Tcl_Interp *interp,
        char *varName));
EXTERN int       Tcl_UnregisterChannel _ANSI_ARGS_((Tcl_Interp *interp,
        Tcl_Channel chan));
EXTERN int       Tcl_UnsetVar _ANSI_ARGS_((Tcl_Interp *interp,
        char *varName, int flags));
EXTERN int       Tcl_UnsetVar2 _ANSI_ARGS_((Tcl_Interp *interp,
        char *part1, char *part2, int flags));
EXTERN void      Tcl_UntraceVar _ANSI_ARGS_((Tcl_Interp *interp,
        char *varName, int flags, Tcl_VarTraceProc *proc,
        ClientData clientData));
EXTERN void      Tcl_UntraceVar2 _ANSI_ARGS_((Tcl_Interp *interp,
        char *part1, char *part2, int flags,

```

```

        Tcl_VarTraceProc *proc, ClientData clientData));
EXTERN void      Tcl_UpdateLinkedVar _ANSI_ARGS_((Tcl_Interp *interp,
        char *varName));
EXTERN int       Tcl_UpVar _ANSI_ARGS_((Tcl_Interp *interp,
        char *frameName, char *varName,
        char *localName, int flags));
EXTERN int       Tcl_UpVar2 _ANSI_ARGS_((Tcl_Interp *interp,
        char *frameName, char *part1, char *part2,
        char *localName, int flags));
EXTERN int       Tcl_VarEval _ANSI_ARGS_((
        TCL_VARARGS(Tcl_Interp *,interp));
EXTERN ClientData Tcl_VarTraceInfo _ANSI_ARGS_((Tcl_Interp *interp,
        char *varName, int flags,
        Tcl_VarTraceProc *procPtr,
        ClientData prevClientData));
EXTERN ClientData Tcl_VarTraceInfo2 _ANSI_ARGS_((Tcl_Interp *interp,
        char *part1, char *part2, int flags,
        Tcl_VarTraceProc *procPtr,
        ClientData prevClientData));
EXTERN int       Tcl_WaitForEvent _ANSI_ARGS_((Tcl_Time *timePtr));
EXTERN Tcl_Pid   Tcl_WaitPid _ANSI_ARGS_((Tcl_Pid pid, int *statPtr,
        int options));
EXTERN int       Tcl_Write _ANSI_ARGS_((Tcl_Channel chan,
        char *s, int slen));
EXTERN void      Tcl_WrongNumArgs _ANSI_ARGS_((Tcl_Interp *interp,
        int objc, Tcl_Obj *CONST objv[], char *message));

```

```

#undef TCL_STORAGE_CLASS
#define TCL_STORAGE_CLASS

```

```

/*
 * Convenience declaration of Tcl_AppInit for backwards compatibility.
 * This function is not *implemented* by the tcl library, so the storage
 * class is neither DLLEXPORT nor DLLIMPORT
 */

```

```

EXTERN int      Tcl_AppInit _ANSI_ARGS_((Tcl_Interp *interp));

```

```

#endif /* RESOURCE_INCLUDED */

```

```

#undef TCL_STORAGE_CLASS
#define TCL_STORAGE_CLASS DLLIMPORT

```

```

#endif /* _TCL */

```

```
/*
 * tkWin.h --
 *
 * Declarations of public types and interfaces that are only
 * available under Windows.
 *
 * Copyright (c) 1996 by Sun Microsystems, Inc.
 *
 * See the file "license.terms" for information on usage and redistribution
 * of this file, and for a DISCLAIMER OF ALL WARRANTIES.
 *
 * RCS: @(#) $Id: tkWin.h,v 1.4 1998/09/14 18:23:59 stanton Exp $
 */
```

```
#ifndef _TKWIN
#define _TKWIN
```

```
#ifndef _TK
#include <tk.h>
#endif
```

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#undef WIN32_LEAN_AND_MEAN
```

```
#ifdef BUILD_tk
# undef TCL_STORAGE_CLASS
# define TCL_STORAGE_CLASS DLLEXPORT
#endif
```

```
/*
 * The following messages are use to communicate between a Tk toplevel
 * and its container window.
 */
```

```
#define TK_CLAIMFOCUS      (WM_USER)
#define TK_GEOMETRYREQ    (WM_USER+1)
#define TK_ATTACHWINDOW   (WM_USER+2)
#define TK_DETACHWINDOW   (WM_USER+3)
```

```

/*
 *-----
 *
 * Exported procedures defined for the Windows platform only.
 *
 *-----
 */

EXTERN Window      Tk_AttachHWND _ANSI_ARGS_((Tk_Window tkwin,
                                              HWND hwnd));
EXTERN HINSTANCE   Tk_GetHINSTANCE _ANSI_ARGS_((void));
EXTERN HWND        Tk_GetHWND _ANSI_ARGS_((Window window));
EXTERN Tk_Window   Tk_HWNDToWindow _ANSI_ARGS_((HWND hwnd));
EXTERN void        Tk_PointerEvent _ANSI_ARGS_((HWND hwnd,
                                              int x, int y));
EXTERN int         Tk_TranslateWinEvent _ANSI_ARGS_((HWND hwnd,
                                              UINT message, WPARAM wParam, LPARAM lParam,
                                              LRESULT *result));

# undef TCL_STORAGE_CLASS
# define TCL_STORAGE_CLASS DLLIMPORT

#endif /* _TKWIN */

```

```

/* -*- Mode: C; tab-width: 4; -*- */
/*****
 * Copyright (c) 1996 Netscape Communications. All rights reserved.
 * Copyright (c) 1997 Lawrence T. Hoff (hoff@bnl.gov). All rights reserved.
 *****/
/*
 * UnixShell.c
 *
 * Netscape Client Plugin API
 * - Function that need to be implemented by plugin developers
 *
 * This file defines a "Template" plugin that plugin developers can use
 * as the basis for a real plugin. This shell just provides empty
 * implementations of all functions that the plugin can implement
 * that will be called by Netscape (the NPP_xxx methods defined in
 * npapi.h).
 *
 */

#include <stdio.h>                                /* sprintf(), fprintf(), etc. */
/*
#include <sys/types.h>                            /* pid_t */
#include <unistd.h>                                /* fork(), system(), unlink(),
etc. */
#include <signal.h>                                /* SIGHUP */
#include <fcntl.h>                                /* O_RDWR */
#include <sys/wait.h>                              /* wait() */
#include "npapi.h"
#include <sys/time.h>                              /* struct timeval */
#include <sys/resource.h>                          /* maximum # of file
descriptors */

/*
** Stuff for the NPP_SetWindow method:
**
#ifdef XP_UNIX
#include <X11/Xlib.h>                                /* Xlib */
#include <X11/Intrinsic.h>                          /* Xt */
#include <X11/StringDefs.h>                          /* "callback" */
#include "InitialI.h"
#include <Xm/Form.h>                                /* Motif layout Widget */
#include <Xm/Label.h>                              /* Motif Picture Widget */
#include <Xm/PushButton.h>                          /* Motif Button Widget */

#include "xpm.h" /* xpm library functions */
#include "ump.xpm" /* UMP in XPM format */

/* buttons in XPM format */
#include "stopup.xpm"
#include "stopdn.xpm"
#include "playup.xpm"
#include "playdn.xpm"
#include "pauseup.xpm"
#include "pausedn.xpm"

extern PerDisplayTablePtr _XtperDisplayList;

```

```

#endif /* XP_UNIX */

/* default library, compatible with TIMIDITY */
static const char *timdir = "/usr/local/lib/timidity";

/* resort to slow sampling rate for busy/slow processors */
static int eightKFlag = 0;

/*****
 *
 *      Stop playing music, kill the subordinate task
 *
 *****/

static void stopMidiPlayer(pid_t pid){
    if(pid != (pid_t) -1) {
        kill(pid, SIGHUP);
        while(waitpid(pid, 0, WNOHANG)>0); /* clean up after dead child */
    }
}

/*****
 *
 *      Function to actually play the music
 *
 *****/

static pid_t startMidiPlayer(const char *filename, int *pipes, int loop, int
start){
    void timiditySetPipe(int p);

    pid_t pid;

    /* set up IPC */
    (void)pipe(pipes);

    printf("HIT startMidiPlayer: %s %xx %d %d \n", filename, *pipes, loop, start);
    fflush(stdout);
    /* if we're the parent, return the child ID */
    pid = fork(); if(pid != (pid_t)0) return pid;

    /* OK, now let's get ourselves somewhat removed from our parent */
    {
        int fd, sig;
        struct rlimit rlp;

        /* start by closing all file descriptors (except stderr, and our pipe) */

```



```

(void) getrlimit(RLIMIT_NOFILE, &rlp);
for(fd = 3; fd < rlp.rlim_max; fd++) if(fd != pipes[0]) (void) close(fd);

/* now attach stdin and sdtout to /dev/null */
(void) freopen("/dev/null", "r", stdin);
(void) freopen("/dev/null", "w", stdout);

/* then remove an netscape-installed signal handlers */
for(sig=0; sig < 64; sig++) signal(sig, SIG_DFL);
}

/* Now play the sound file */

{
    static const char *argv[4];
    argv[0] = "timidity";
    argv[1] = "-Od";
    argv[2] = eightKFlag ? "-s8000": "-s22050"; /* check if the user specified
8K rate */
    argv[3] = filename;

    /* change to the configuration directory */
    chdir(timdir);

    /* tell timidity where to read GUI input */
    timiditySetPipe(pipes[0]);

    /* loop forever, until unloaded, wait for play to be enabled */
    do {

        signed char c;

        /* wait for a new play command before starting */
        if(!start) while (read(pipes[0], &c, 1) == 1 && c != ' ');

        /* fprintf(stderr, "GOT START: %d\n", start); fflush(stderr); */
        start = -0; /* we've started now! */

        /* fprintf(stderr, "CALL: %s %s %s %s\n", argv[0], argv[1], argv[2], argv[3]);
        fflush(stderr); */
        /* jump to timidity */
        {
            int status;
            int timiditymain();
        /* fprintf(stderr, "timiditymain: %xx ME: %d \n", timiditymain, getpid());
        fflush(stderr); */
            status = timiditymain(4, argv);
        /* fprintf(stderr, "back from main %d \n", status); fflush(stderr); */
        }

        if(!loop){
            /* wait for a new stop command before starting again */
            while (read(pipes[0], &c, 1) == 1 && c != ' ');

            /* wait for a new play command before starting again */
            while (read(pipes[0], &c, 1) == 1 && c != ' ');
        }
    }
}

```

```

    } while(~0); /* loop forever */

}

/* bye now */
_exit(0);

}

/*****
 * Instance state information about the plugin.
 *
 * PLUGIN DEVELOPERS:
 *   Use this struct to hold per-instance information that you'll
 *   need in the various functions in this file.
 *****/

/* we'll read in all the file before processing. Typical MIDI files are
smallish,
so it should be OK */
static const char *filestub = "/tmp/midi";

typedef enum { PLUGIN_PLAYING, PLUGIN_STOPPED, PLUGIN_PAUSED } pluginPlayState;

typedef struct _PluginInstance
{
    int fd; /* file descriptor for temp file */
    const char *filename; /* name of same */
    pid_t pid; /* process ID of sound player (FIXME -- should be global???) */
    int pipes[2]; /* inter-process communication */
    pluginPlayState pluginState; /* stopped, paused, or playing */
    int loop; /* 0 means play only once, non-zero means loop forever */

    /* UNIX data members */
#ifdef XP_UNIX
    Widget play, pause, stop; /* button widgets, and their pictures, as well as
the logo picture */
    Pixmap logoPixmap, playupPixmap, playdnPixmap, pauseupPixmap, pausednPixmap,
stopupPixmap, stopdnPixmap ;
    Display *display; /* needed for XFreePixmap() */
    int width, height; /* how big is the UMP logo? */
#endif /* XP_UNIX */

    FILE *tst;
} PluginInstance;

/*****
 *
 * Empty implementations of plugin API functions
 *
 * PLUGIN DEVELOPERS:
 *   You will need to implement these functions as required by your
 *   plugin.
 *****/

```

```

char*
NPP_GetMIMEDescription(void)
{
    static char *desc =
        "audio/midi:mid,midi,foo:<A HREF=mailto:hoff@bnl.gov>Larry Hoff's</A> "
        "UMP plugin version 1.10;"
        "audio/x-midi:mid,midi,bar:<A HREF=mailto:hoff@bnl.gov>Larry Hoff's</A> "
        "UMP plugin version 1.10;";

    return(desc);
}

NPPError
NPP_GetValue(void *future, NPPVariable variable, void *value)
{
    NPPError err = NPERR_NO_ERROR;

    static char *desc =

        "This plugins plays MIDI, using the timidity "
        "toolkit, via the standard audio device.<P>"
        "By default, timidity configuration files are in the directory "
        "/usr/local/lib/timidity<P>"
        "Use the TIMID_DIR environment variable to change "
        "the configuration directory.<P>"
        "You may set the environment variable TIMID_8K to "
        "reduce the sampling rate (to 8000 Hz) for busy or slow "
        "processors, or for machines which only support 8KHz.<P>";

    switch (variable) {
    case NPPVpluginNameString:
        *((char **)value) = "UNIX MIDI plugin";
        break;
    case NPPVpluginDescriptionString:
        *((char **)value) = desc;
        break;
    default:
        err = NPERR_GENERIC_ERROR;
    }
    return err;
}

NPPError
NPP_Initialize(void)
{
    /* see if the user has specified an alternate directory for timidity */
    if(getenv("TIMID_DIR") != NULL) timdir = getenv("TIMID_DIR");

    /* see if the user has specified an alternate directory for timidity */
    if(getenv("TIMID_8K") != NULL) eightKFlag = 1;

    return NPERR_NO_ERROR;
}

```

```
}
```

```
NPErr
```

```
NPP_New(NPMIMEType pluginType,  
        NPP instance,  
        uint16 mode, /* either NP_EMBED or NP_FULL, we don't care */  
        int16 argc, /* number of name/value pairs */  
        char* argn[],  
        char* argv[],  
        NPSavedData* saved)
```

```
{
```

```
    PluginInstance* This;
```

```
    if (instance == NULL)  
        return NPERR_INVALID_INSTANCE_ERROR;
```

```
    instance->pdata = NPN_MemAlloc(sizeof(PluginInstance));
```

```
    This = (PluginInstance*) instance->pdata;
```

```
    if (This == NULL)  
        return NPERR_OUT_OF_MEMORY_ERROR;
```

```
    fprintf(stderr, "11 INSTANCE: %xx\n", instance); fflush(stderr);
```

```
{
```

```
    /* construct a filename unique to this instance to  
       store the data in */
```

```
    static char buf[256];  
    sprintf(buf, "%s%x.mid", filestub, This);
```

```
    This->filename = buf;
```

```
    This->pid = -1;
```

```
    This->pipes[1] = -1;
```

```
    This->pipes[0] = -1;
```

```
    This->logoPixmap = -1;
```

```
    This->playupPixmap = -1;
```

```
    This->playdnPixmap = -1;
```

```
    This->pauseupPixmap = -1;
```

```
    This->pausednPixmap = -1;
```

```
    This->stopupPixmap = -1;
```

```
    This->stopdnPixmap = -1;
```

```
    This->pluginState = PLUGIN_PLAYING;
```

```
    This->loop = 0;
```

```
        This->tst = fopen("/tmp/log", "w");
```

```
        fprintf(This->tst, "New\n");
```

```
        fflush(This->tst);
```

```
}
```

```
/* fprintf(stderr, "22INSTANCE: %xx\n", instance); fflush(stderr); */
```

```
{
```

```
    /* handle user-overrides */
```

```
    int i;
```

```

    for(i = 0; i < argc; i++) {
        if(strcasecmp(argn[i], "loop") == 0){
            if(strcasecmp(argv[i], "true") == 0 ||
                strcmp(argv[i], "yes") == 0) {
                This->loop = ~0;
            }
        }
        else if(strcasecmp(argn[i], "autostart") == 0){
            if(strcasecmp(argv[i], "false") == 0 ||
                strcmp(argv[i], "no") == 0) {
                This->pluginState = PLUGIN_STOPPED;
            }
        }
    }
}

/* fprintf(stderr, "33 INSTANCE: %xx\n", instance); fflush(stderr); */

/* Load associated JAVA class */
(void) setupLiveConnect(instance, This);
/* fprintf(stderr, "44 INSTANCE: %xx\n", instance); fflush(stderr); */

return NPERR_NO_ERROR;
}

NPErrors
NPP_Destroy(NPP instance, NPSaveData** save)
{
    PluginInstance* This;

    if (instance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;

    This = (PluginInstance*) instance->pdata;

    /* PLUGIN DEVELOPERS:
     * If desired, call NP_MemAlloc to create a
     * NPSaveData structure containing any state information
     * that you want restored if this plugin instance is later
     * recreated.
     */

    /* quiet things up ASAP */
    if(This->pluginState == PLUGIN_PLAYING){

        /* ' ' means toggle pause */
        write(This->pipes[1], " ", 1);
    }

    if (This != NULL) {
        /* get rid of the temporary file */
        (void) unlink(This->filename);

        /* clean up IPC */
    }
}

```

```

        close(This->pipes[1]);
        close(This->pipes[0]);

#ifdef XP_UNIX
        if(This->logoPixmap != -1)XFreePixmap(This->display, This->logoPixmap);
        if(This->playupPixmap != -1)XFreePixmap(This->display, This->playupPixmap);
        if(This->playdnPixmap != -1)XFreePixmap(This->display, This->playdnPixmap);
        if(This->pauseupPixmap != -1)XFreePixmap(This->display, This->
>pauseupPixmap);
        if(This->pausednPixmap != -1)XFreePixmap(This->display, This->
>pausednPixmap);
        if(This->stopupPixmap != -1)XFreePixmap(This->display, This->stopupPixmap);
        if(This->stopdnPixmap != -1)XFreePixmap(This->display, This->stopdnPixmap);
#endif

        /* stop player */
        stopMidiPlayer(This->pid);

        NPN_MemFree(instance->pdata);
        instance->pdata = NULL;
    }

    return NPERR_NO_ERROR;
}

/* give buttons the right look for the state of the plugin */

static void setButtons(PluginInstance* This, pluginPlayState state){

    switch(This->pluginState = state){

        case PLUGIN_PLAYING :
            if(This->playdnPixmap != -1)
                XtVaSetValues(This->play, XmNlabelType, XmPIXMAP, XmNlabelPixmap, This->
>playdnPixmap, NULL);
            if(This->pauseupPixmap != -1)
                XtVaSetValues(This->pause, XmNlabelType, XmPIXMAP, XmNlabelPixmap, This->
>pauseupPixmap, NULL);
            if(This->stopupPixmap != -1)
                XtVaSetValues(This->stop, XmNlabelType, XmPIXMAP, XmNlabelPixmap, This->
>stopupPixmap, NULL);
            break;

        case PLUGIN_PAUSED :
            if(This->playupPixmap != -1)
                XtVaSetValues(This->play, XmNlabelType, XmPIXMAP, XmNlabelPixmap, This->
>playdnPixmap, NULL);
            if(This->pausednPixmap != -1)
                XtVaSetValues(This->pause, XmNlabelType, XmPIXMAP, XmNlabelPixmap, This->
>pausednPixmap, NULL);
            if(This->stopupPixmap != -1)
                XtVaSetValues(This->stop, XmNlabelType, XmPIXMAP, XmNlabelPixmap, This->
>stopupPixmap, NULL);
            break;

        case PLUGIN_STOPPED :
            if(This->playupPixmap != -1)

```

```

        XtVaSetValues(This->play, XmNlabelType, XmPIXMAP, XmNlabelPixmap, This->playupPixmap, NULL);
        if(This->pauseupPixmap != -1)
            XtVaSetValues(This->pause, XmNlabelType, XmPIXMAP, XmNlabelPixmap, This->pauseupPixmap, NULL);
        if(This->stopdnPixmap != -1)
            XtVaSetValues(This->stop, XmNlabelType, XmPIXMAP, XmNlabelPixmap, This->stopdnPixmap, NULL);
        break;
    }
}

```

/* handle state changes caused by button pushes */

```

static void pauseSelected(Widget w, XtPointer client_data, XtPointer call_data){
    PluginInstance *This = (PluginInstance *) client_data;

    if(This->pluginState != PLUGIN_STOPPED){

        /* ' ' means toggle pause */
        write(This->pipes[1], " ", 1);

        setButtons(This, This->pluginState == PLUGIN_PAUSED? PLUGIN_PLAYING:
        PLUGIN_PAUSED);
    }
}

```

```

static void playSelected(Widget w, XtPointer client_data, XtPointer call_data){
    PluginInstance *This = (PluginInstance *) client_data;

    if(This->pluginState != PLUGIN_PLAYING){

        /* ' ' means toggle pause */
        write(This->pipes[1], " ", 1);

        setButtons(This, PLUGIN_PLAYING);
    }
}

```

```

static void stopSelected(Widget w, XtPointer client_data, XtPointer call_data){
    PluginInstance *This = (PluginInstance *) client_data;

    if(This->pluginState == PLUGIN_PAUSED){

        /* ' q ' means toggle pause, quit, then toggle pause */
        write(This->pipes[1], " q ", 3);

        setButtons(This, PLUGIN_STOPPED);
    }
}

```

```

    }
    else if(This->pluginState == PLUGIN_PLAYING){

        /* 'q ' means quit, then toggle pause */
        write(This->pipes[1], "q ", 2);

        setButtons(This, PLUGIN_STOPPED);
    }
}

static void writePlayStatus(Widget w, XtPointer closure, XEvent* event, Boolean*
b){

    NPN_Status((NPP)closure, "Play MIDI music");
    *b = False;

}

static void writePauseStatus(Widget w, XtPointer closure, XEvent* event,
Boolean* b){

    NPN_Status((NPP)closure, "Pause/Resume MIDI music");
    *b = False;

}

static void writeStopStatus(Widget w, XtPointer closure, XEvent* event, Boolean*
b){

    NPN_Status((NPP)closure, "Stop MIDI music");
    *b = False;

}

static void clearStatus(Widget w, XtPointer closure, XEvent* event, Boolean* b){

    NPN_Status((NPP)closure, "");
    *b = False;

}

NPErr
NPP_SetWindow(NPP instance, NPWindow* window)
{
    PluginInstance* This;

    if (instance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;

    if (window == NULL)
        return NPERR_NO_ERROR;

    This = (PluginInstance*) instance->pdata;

```



```

fprintf(This->tst, "SetWindow\n");fflush(This->tst);

/*
 * PLUGIN DEVELOPERS:
 * Before setting window to point to the
 * new window, you may wish to compare the new window
 * info to the previous window (if any) to note window
 * size changes, etc.
 */

#ifdef XP_UNIX
fprintf(This->tst, "setCallBack\n");fflush(This->tst);
{

    Widget netscape_widget, form, logo;
    XpmAttributes attr;

    NPSetWindowCallbackStruct *winInfo = (NPSetWindowCallbackStruct *)window-
>ws_info;
    fprintf(This->tst, "post setCallBack: winInfo: %xx\n", winInfo);fflush(This-
>tst);
    fprintf(This->tst, "post setCallBack: display: %xx\n", winInfo-
>display);fflush(This->tst);
    fprintf(This->tst, "window: %xx x: %xx y: %xx width: %xx height: %xx clipRect
%xx ws-info: %xx\n",
        window->window, window->x, window->y, window->width, window->height, window-
>clipRect, window->ws_info); fflush(This->tst);

    This->display = winInfo->display;
    fprintf(This->tst, "done display\n");fflush(This->tst);

    /* tell XPM to allocate fairly close (not exact) colors from the window's
    colormap */
    attr.valuemask =
XpmSize|XpmCloseness|XpmAllocCloseColors|XpmExactColors|XpmColormap|XpmDepth|Xpm
Visual;
    attr.closeness = 30000;
    attr.alloc_close_colors = True;
    attr.exactColors = False;
    attr.colormap = winInfo->colormap;
    attr.depth = winInfo->depth;
    attr.visual = winInfo->visual;
    fprintf(This->tst, "colormap: %xx depth: %xx visual: %xx type: %xx\n",
attr.colormap, attr.depth, attr.visual, winInfo->type); fflush(This->tst);

    /* while it's at it, tell us what size the logo is */
    /* reuse the pixmap (if possible), assume the colormap and depth will always
    be the same */
    /* otherwise we'd have to call XFreePixmap in the XtNdestroyCallback for
    each widget */
    if(This->playupPixmap == -1) {
        if(XpmCreatePixmapFromData(This->display, (Window) window->window, ump,
&This->logoPixmap, NULL, &attr) == XpmSuccess) {

```

```

        This->width = attr.width;
        This->height = attr.height;
    }
}

fprintf(This->tst, "create Pixmap 1\n");fflush(This->tst);
if(This->playupPixmap == -1)
    (void)XpmCreatePixmapFromData(This->display, (Window) window->window,
playup, &This->playupPixmap, NULL, &attr);

fprintf(This->tst, "create Pixmap 2\n");fflush(This->tst);
if(This->playdnPixmap == -1)
    (void)XpmCreatePixmapFromData(This->display, (Window) window->window,
playdn, &This->playdnPixmap, NULL, &attr);

fprintf(This->tst, "create Pixmap 3\n");fflush(This->tst);
if(This->pauseupPixmap == -1)
    (void)XpmCreatePixmapFromData(This->display, (Window) window->window,
pauseup, &This->pauseupPixmap, NULL, &attr);

if(This->pausednPixmap == -1)
    (void)XpmCreatePixmapFromData(This->display, (Window) window->window,
pausedn, &This->pausednPixmap, NULL, &attr);

if(This->stopupPixmap == -1)
    (void)XpmCreatePixmapFromData(This->display, (Window) window->window,
stopup, &This->stopupPixmap, NULL, &attr);

if(This->stopdnPixmap == -1)
    (void)XpmCreatePixmapFromData(This->display, (Window) window->window,
stopdn, &This->stopdnPixmap, NULL, &attr);

/* determine our parentage */
{
    Display *disp;
    disp = This->display;

    fprintf(This->tst, "Disp Ht: %d Disp Wd: %d \n", DisplayHeight(disp, 0),
DisplayWidth(disp, 0));
}
fprintf(This->tst, "parentage %xx %xx displaylist: %xx\n", This->display,
(Window) window->window, _XtperDisplayList); fflush(This->tst);

netscape_widget = XtWindowToWidget(This->display, (Window) window->window);

/* fit our widgets into a form */
/* no spacing between widgets, we use pixmaps in each widget to define
borders */
fprintf(This->tst, "UMP form %xx %xx\n", xmFormWidgetClass, netscape_widget);
fflush(This->tst);
form = XtVaCreateWidget("UMP form", xmFormWidgetClass, netscape_widget,
NULL);

```

```

/* Some buttons */
fprintf(This->tst, "Managed Widget\n");fflush(This->tst);
This->stop = XtVaCreateManagedWidget("stop", xmPushButtonWidgetClass, form,
    XmNshadowThickness, 0,
    XmNhighlightThickness, 0,
    XmNmarginHeight, 0,
    XmNmarginWidth, 0,
    XmNtopAttachment, XmATTACH_FORM,
    XmNleftAttachment, XmATTACH_FORM,
    XmNtopOffset, 0,
    XmNleftOffset, 28,
    NULL);

This->pause = XtVaCreateManagedWidget("pause", xmPushButtonWidgetClass,
form,
    XmNshadowThickness, 0,
    XmNhighlightThickness, 0,
    XmNmarginHeight, 0,
    XmNmarginWidth, 0,
    XmNtopAttachment, XmATTACH_FORM,
    XmNleftAttachment, XmATTACH_WIDGET,
    XmNleftWidget, This->stop,
    XmNtopOffset, 0,
    XmNleftOffset, 6,
    NULL);

This->play = XtVaCreateManagedWidget("play", xmPushButtonWidgetClass, form,
    XmNshadowThickness, 0,
    XmNhighlightThickness, 0,
    XmNmarginHeight, 0,
    XmNmarginWidth, 0,
    XmNtopAttachment, XmATTACH_FORM,
    XmNleftAttachment, XmATTACH_WIDGET,
    XmNleftWidget, This->pause,
    XmNtopOffset, 0,
    XmNleftOffset, 7,
    NULL);

/* tell us when the buttons are pushed */
XtAddCallback(This->play, XmNactivateCallback, playSelected, (XtPointer)
This);
XtAddCallback(This->pause, XmNactivateCallback, pauseSelected, (XtPointer)
This);
XtAddCallback(This->stop, XmNactivateCallback, stopSelected, (XtPointer)
This);

/* write help text when the mouse is over the button */
XtAddEventHandler(This->play, EnterWindowMask, False, writePlayStatus,
(XtPointer) instance);
XtAddEventHandler(This->play, LeaveWindowMask, False, clearStatus,
(XtPointer) instance);

XtAddEventHandler(This->pause, EnterWindowMask, False, writePauseStatus,
(XtPointer) instance);
XtAddEventHandler(This->pause, LeaveWindowMask, False, clearStatus,
(XtPointer) instance);

```

```

    XtAddEventHandler(This->stop, EnterWindowMask, False, writeStopStatus,
(XtPointer) instance);
    XtAddEventHandler(This->stop, LeaveWindowMask, False, clearStatus,
(XtPointer) instance);

```

```

/* Here's our logo, no border around our nice pixmap */
logo = XtVaCreateManagedWidget("UMP logo", xmLabelWidgetClass, form,
                                XmNshadowThickness, 0,
                                XmNhighlightThickness, 0,
                                XmNmarginHeight, 0,
                                XmNmarginWidth, 0,
                                XmNtopAttachment, XmATTACH_FORM,
                                XmNleftAttachment, XmATTACH_FORM,
                                XmNtopOffset, 0,
                                XmNleftOffset, 0,
                                NULL);

```

```

/* show the logo */
if(This->logoPixmap != -1)
    XtVaSetValues(logo, XmNlabelType, XmPIXMAP, XmNlabelPixmap, This-
>logoPixmap, NULL);

```

```

/* give buttons the right look and feel */
setButtons(This, This->pluginState);

```

```

/* hand over control. Our job is done! */
XtManageChild(form);

```

```

/* jam it to the correct size, otherwise Netscape lets us go wild! */
XtMakeResizeRequest(form,
                    This->width<window->width?This->width:window->width,
                    This->height<window->height?This->height:window->height,
                    NULL, NULL);

```

```

}

```

```

#endif /* XP_UNIX */

```

```

    return NPERR_NO_ERROR;
}

```

NPError

```

NPP_NewStream(NPP instance,
              NPMIMETYPE type,
              NPStream *stream,
              NPBool seekable,
              uint16 *styp)

```

```

{

```

```

    NPByteRange range;
    PluginInstance* This;

```

```

    if (instance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;

```

```

    This = (PluginInstance*) instance->pdata;

```

```

/* open the file to store data in */
This->fd = open(This->filename, O_RDWR|O_CREAT, 0666);
if(This->fd == -1) return NPERR_GENERIC_ERROR;

return NPERR_NO_ERROR;
}

/* PLUGIN DEVELOPERS:
 *   These next 2 functions are directly relevant in a plug-in which
 *   handles the data in a streaming manner. If you want zero bytes
 *   because no buffer space is YET available, return 0. As long as
 *   the stream has not been written to the plugin, Navigator will
 *   continue trying to send bytes. If the plugin doesn't want them,
 *   just return some large number from NPP_WriteReady(), and
 *   ignore them in NPP_Write(). For a NP_ASFILE stream, they are
 *   still called but can safely be ignored using this strategy.
 */

int32 STREAMBUFSIZE = 0X0FFFFFFF; /* If we are reading from a file in NPAsFile
                                     * mode so we can take any size stream in our
                                     * write call (since we ignore it) */

int32
NPP_WriteReady(NPP instance, NPStream *stream)
{
    PluginInstance* This;
    if (instance != NULL)
        This = (PluginInstance*) instance->pdata;

    return STREAMBUFSIZE;
}

int32
NPP_Write(NPP instance, NPStream *stream, int32 offset, int32 len, void *buffer)
{
    if (instance != NULL)
    {
        PluginInstance* This = (PluginInstance*) instance->pdata;
        len = write(This->fd, buffer, len);
        if(len < 0) close(This->fd);
    }

    return len;          /* The number of bytes accepted */
}

NPError
NPP_DestroyStream(NPP instance, NPStream *stream, NPError reason)
{
    PluginInstance* This;

    if (instance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;
    This = (PluginInstance*) instance->pdata;

```

```

close(This->fd);

/* stop another player in progress ???*/
stopMidiPlayer(This->pid);

/* yahoo, now we have the data, run timidity */
This->pid = startMidiPlayer(This->filename, This->pipes, This->loop, This-
>pluginState == PLUGIN_PLAYING);

NPN_Status(instance, "Playing MIDI file");

return NPERR_NO_ERROR;
}

void
NPP_StreamAsFile(NPP instance, NPStream *stream, const char* fname)
{
    PluginInstance* This;
    if (instance != NULL)
        This = (PluginInstance*) instance->pdata;

    /* just spawn timidity */
    This->pid = startMidiPlayer(fname, This->pipes, This->loop, This->pluginState
== PLUGIN_PLAYING);

    NPN_Status(instance, "Playing MIDI file");
}

void
NPP_Print(NPP instance, NPPrint* printInfo)
{
    if (printInfo == NULL)
        return;

    if (instance != NULL) {
        PluginInstance* This = (PluginInstance*) instance->pdata;

        if (printInfo->mode == NP_FULL) {
            /*
             * PLUGIN DEVELOPERS:
             *   If your plugin would like to take over
             *   printing completely when it is in full-screen mode,
             *   set printInfo->pluginPrinted to TRUE and print your
             *   plugin as you see fit. If your plugin wants Netscape
             *   to handle printing in this case, set
             *   printInfo->pluginPrinted to FALSE (the default) and
             *   do nothing. If you do want to handle printing
             *   yourself, printOne is true if the print button
             *   (as opposed to the print menu) was clicked.
             *   On the Macintosh, platformPrint is a THPrint; on
             *   Windows, platformPrint is a structure
             *   (defined in npapi.h) containing the printer name, port,
             *   etc.
            */

```

```

    */

    void* platformPrint =
    printInfo->print.fullPrint.platformPrint;
    NPBool printOne =
    printInfo->print.fullPrint.printOne;

    /* Do the default*/
    printInfo->print.fullPrint.pluginPrinted = FALSE;
}
else { /* If not fullscreen, we must be embedded */
    /*
    * PLUGIN DEVELOPERS:
    *   If your plugin is embedded, or is full-screen
    *   but you returned false in pluginPrinted above, NPP_Print
    *   will be called with mode == NP_EMBED. The NPWindow
    *   in the printInfo gives the location and dimensions of
    *   the embedded plugin on the printed page. On the
    *   Macintosh, platformPrint is the printer port; on
    *   Windows, platformPrint is the handle to the printing
    *   device context.
    */

    NPWindow* printWindow =
    &(printInfo->print.embedPrint.window);
    void* platformPrint =
    printInfo->print.embedPrint.platformPrint;
}
}

#ifdef INCLUDE_LIVE_CONNECT

setupLiveConnect(NPP instance, PluginInstance *This){}

jref
NPP_GetJavaClass()
{
    return NULL;
}

void
NPP_Shutdown(void)
{
}

#else

/* IMHO, Live Connect adds lots of overhead and complexity compared to
the extra functionality it provides (for UMP, anyway). Here we
allow Java applets or JavaScript to operate the stop/pause/play
buttons. For this, we must load the "UMP" java class file, which
must be in the user's CLASSPATH. Then we cannot even operate
directly on the Widgets, because X gets confused when run from an
applet thread, so we use a short (msec) timeout which actually
does the work.
*/

```

```

#include "UMP.c" /* created by : javac UMP.java; javah -jri -stubs UMP */

setupLiveConnect(NPP instance, PluginInstance *This){

    /* store the pointer-to-object for later use (one Java peer per plugin
instance) */
    UMP_setThis(NPN_GetJavaEnv(), NPN_GetJavaPeer(instance), (int) This);

}

jref
NPP_GetJavaClass()
{
    /* enable Live Connect */
    return use_UMP(NPN_GetJavaEnv());
}

void
NPP_Shutdown(void)
{
    /* all done with Live Connect */
    unuse_UMP(NPN_GetJavaEnv());
}

/* functions for LiveConnect */

static void stopSelectedCallback(XtPointer This, XtIntervalId* id){
stopSelected(NULL, This, NULL); }

void native_UMP_stopSelected(JRIEnv* env, UMP *self, jint This){

    PluginInstance *Ump = (PluginInstance *)This;

    (void) XtAppAddTimeOut(XtWidgetToApplicationContext(Ump->stop), 1,
stopSelectedCallback, (XtPointer) This);

}

static void playSelectedCallback(XtPointer This, XtIntervalId* id){
playSelected(NULL, This, NULL); }

/* functions for LiveConnect */
void native_UMP_playSelected(JRIEnv* env, UMP *self, jint This){

    PluginInstance *Ump = (PluginInstance *)This;

    (void) XtAppAddTimeOut(XtWidgetToApplicationContext(Ump->stop), 1,
playSelectedCallback, (XtPointer) This);

}

static void pauseSelectedCallback(XtPointer This, XtIntervalId* id){
pauseSelected(NULL, This, NULL); }

```



```
/* functions for LiveConnect */
void native_UMP_pauseSelected(JRIEnv* env, UMP *self, jint This){

    PluginInstance *Ump = (PluginInstance *)This;

    (void) XtAppAddTimeOut(XtWidgetToApplicationContext(Ump->stop), 1,
        pauseSelectedCallback, (XtPointer) This);
}

#endif
```

```

/*
 * plugext1.0 is a compiled version of plugext for Tcl.
 * -- Clif Flynt <clif@cflynt.com> Jan 3, 1999
 * --
 * -- Modified from get.c by
 * -- Jean-Claude Wippler <jcw@equi4.com>, September 17, 1998.
 */

#include <stdio.h>
#ifdef (_WIN32)
#include <windows.h>
#else
#include <dlfcn.h>
#endif
#include <tcl.h>

Tcl_HashTable *dlopen_hashtablePtr;
extern int dummyDebugPrint;

void dlopen_InitHashTable () {
    dlopen_hashtablePtr = (Tcl_HashTable *) ckalloc(sizeof(Tcl_HashTable));
    Tcl_InitHashTable(dlopen_hashtablePtr, TCL_STRING_KEYS);
}

static int
dlopen(ClientData x, Tcl_Interp* interp, int objc, Tcl_Obj*CONST* objv)
{
    char hash_handle[50];
    char *tmp;
    void *handle;
    Tcl_HashEntry *hashEntryPtr;
    Tcl_Obj *returnObjPtr;
    int isNew;

    if (objc > 1)
    {
        tmp = Tcl_GetStringFromObj(objv[1], NULL);
#ifdef (_WIN32)
        handle = LoadLibrary(tmp);
#else
        handle = dlopen(tmp, RTLD_NOW | RTLD_GLOBAL);
#endif

        if (!handle) {
            fprintf(stderr, "FAIL IN DLOPEN: %s\n", tmp);
#ifdef (_WIN32)
            fprintf(stderr, "%s\n", "Is there a windows error fcn");
#else
            fprintf(stderr, "%s\n", dlerror());
#endif
            return(TCL_ERROR);
        }
    }
}

```

```

/*
 * Allocate the space and initialize the return structure.
 */

sprintf(hash_handle, "X%x", handle);

hashEntryPtr = Tcl_CreateHashEntry(dlopen_hashtablePtr, hash_handle,
&isNew);
Tcl_SetHashValue(hashEntryPtr, handle);

returnObjPtr = Tcl_NewStringObj(hash_handle, -1);

Tcl_SetObjResult(interp, returnObjPtr);

return TCL_OK;
}

__getfpucw() {printf("hit getfpucw\n"); fflush(stdout);}

#if defined (_WIN32)

static int
dlSym(ClientData x, Tcl_Interp* interp, int objc, Tcl_Obj*CONST* objv) {

    Tcl_Obj *returnObjPtr;

    returnObjPtr = Tcl_NewStringObj("dlSym not implemented for Windows", -1);

    Tcl_SetObjResult(interp, returnObjPtr);
    return TCL_OK;
}

#else

static int
dlSym(ClientData x, Tcl_Interp* interp, int objc, Tcl_Obj*CONST* objv)
{
    char *hash_handle;
    char *tmp;
    void *handle;
    Tcl_HashEntry *hashEntryPtr;
    Tcl_Obj *returnObjPtr;
    void *locat;
    char rtnString[20];

    if (objc > 1)
    {
        hash_handle = Tcl_GetStringFromObj(objv[1], NULL);
        hashEntryPtr = Tcl_FindHashEntry(dlopen_hashtablePtr, hash_handle);
        if (hashEntryPtr == (Tcl_HashEntry *) NULL) {
            char errString[80];
            Tcl_Obj *errCodePtr;

            /*

```

```

    * Define an error code from an integer, and set errorCode.
    */
    errCodePtr = Tcl_NewIntObj(554);
    Tcl_SetObjErrorCode(interp, errCodePtr);

    /*
    * This string will be placed in the global variable errorInfo
    */

    sprintf(errString, "Hash object \"%s\" does not exist.", hash_handle);
    Tcl_AddErrorInfo(interp, errString);

    /*
    * This string will be returned as the result of the command.
    */
    Tcl_AppendResult(interp, "can not find hashed object named \"",
        hash_handle, "\"", (char *) NULL);
    return TCL_ERROR;
}

handle = (char *)Tcl_GetHashValue(hashEntryPtr);

tmp = Tcl_GetStringFromObj(objv[2], NULL);

locat = dlsym(handle, tmp);

sprintf(rtnString, "%x", locat);

}

/*
* Allocate the space and initialize the return structure.
*/

returnObjPtr = Tcl_NewStringObj(rtnString, -1);

Tcl_SetObjResult(interp, returnObjPtr);

return TCL_OK;
}
#endif

#if defined (_WIN32)

    #define WINEXPORT(t) __declspec(dllexport) t
#elif defined (_WIN)
    #define WINEXPORT(t) t _export
#else
    #define WINEXPORT(t) t
#endif

#if defined (__MWERKS__)
    #pragma export on
#endif

```

```
WINEXPORT(int) Dlopen_Init(Tcl_Interp* interp)
{
    Tcl_CreateObjCommand(interp, "dlopen", dlOpen, 0, 0);
    Tcl_CreateObjCommand(interp, "dlsym", dlSym, 0, 0);

    dlopen_InitHashTable ();

    Tcl_PkgProvide(interp, "dlopen", "1.0");

    return TCL_OK;
}

WINEXPORT(int) Dlopen_SafeInit(Tcl_Interp* interp)
{
    return Dlopen_Init(interp);
}
```

```
package require plugext
```

```
#
#
# Embed - state variable
# Embed(uniq) - A unique number for the instances
# Embed(UPPER.$id) - A keyword/value pair for an instance.
# Embed(lower.$id) - A value assigned by the embed command.
#
# Embed(MIMETYPE) - The command for this creation of the plugin
# Embed(instance.$id) - The command for this instance of the plugin
# Embed(canvas.$id) - The canvas for this instance of the plugin
#
# Adds one new arg to embed xx=yy -
# KEEPOPEN=1 default - the embed call does not close the stream after
# sending data to the plugin.
# KEEPOPEN=0 default - the embed call closes the stream after
# sending data to the plugin. Closing the stream causes some
# plugins to start.
```

```
namespace eval EMBED {
variable Embed
variable output [open argh w]
```

```
set Embed(uniq) 1
set Embed(CanvasParent) ""
```

```
switch $tcl_platform(platform) {
  "unix" {
    package require dlopen
    set Embed(xref) {
      { .mid audio/x-midi /usr/lib/netscape/plugins/ump.so }
      { .midi audio/x-midi /usr/lib/netscape/plugins/ump.so }
    }
  }
  "windows" {
    set Embed(xref) {
      { .rpm audio/x-pn-realaudio-plugin ./nppl3260.dll }
      { .rm audio/x-pn-realaudio-plugin ./nppl3260.dll }
      { .spl application/futuresplash ./npswf32.dll }
      { .swf application/shockwave-flash ./npswf32.dll }
      { .pdf application/pdf ./nppdf32.dll }
      { .afl video/animaflex ./nprub.dll }
      { .ivf video/x-ivf ./npindeo.dll }
      { .wav audio/x-wav ./npwave.dll }
      { .mid audio/x-midi ./midid.dll }
      { .midi audio/x-midi ./midid.dll }
      { .npw application/x-npwrap ./NPWrap.dll }
      { .mid audio/x-midi ./npmidi.dll }
      { .midi audio/x-midi ./npmidi.dll }
    }
  }
}
```

```
proc getCanvas {id} {
  variable Embed
  if {[info exists Embed(canvas.$id)]} {
```

```

        return $Embed(canvas.$id)
    } else {
        return ""
    }
}

proc getInstance {id} {
    variable Embed
    if {[info exists Embed(canvas.$id)]} {
        return $Embed(instance.$id)
    } else {
        return ""
    }
}

proc setCanvasParent {windowID} {
    set Embed(CanvasParent) $windowID
}

proc embed {args} {
    global tcl_platform
    variable Embed
    variable output

    set id $Embed(uniq)
    incr Embed(uniq)

    set mode "EMBED"
    set Embed(HIDDEN.$id) true
    set Embed(KEEPOPEN.$id) 1

    #
    # Parse the args.
    # Find the key=val and key="val" pairs and convert them to
    # set Embed(KEY) val
    # commands.
    #

    foreach arg $args {
        if {[string first "=" $arg] < 0} {continue}
        foreach {key val} [split $arg "="] {
            set val [string trim $val {""]}
            set key [string toupper $key]
        }
        eval [list set Embed($key.$id) $val]
    }

    #
    # If the plugext extension is not loaded, load it.
    # If *n[ui]x platform, dlopen the required X libraries.
    #

    if {[string match "" [info command plugext]]} {
        if {[string match $tcl_platform(platform) unix]} {
            puts $output "loading dlopen, etc" ; flush $output
            # load "./dlopen"

```

```

        dlopen "/usr/X11R6/lib/libXt.so"
        dlopen "/usr/local/lib/libXm.so"
    }
    # load plugext
    plugext debug 1
}

#
# Figure out which plugin we need. Start by checking
# the SRC=...ext to see if we've got that extension in our list.
#

if {[info exists Embed(SRC.$id)]} {
    set ext [file extension $Embed(SRC.$id)]
    set pos [lsearch $Embed(xref) *${ext}*]
    if {$pos >= 0} {
        set pluginPath [lindex [lindex $Embed(xref) $pos] 2]
        if {[info exists Embed(TYPE.$id)]} {
            set Embed(TYPE.$id) [lindex [lindex $Embed(xref) $pos] 1]
        }
    }
}

# If no SRC matched, see if we can get the required info
# via the Mime TYPE=xx value

if {[info exists pluginPath] && [info exists Embed(TYPE.$id)]} {
    set pos [lsearch $Embed(xref) *$Embed(TYPE.$id)*]
    if {$pos < 0} {break}
    set pluginPath [lindex [lindex $Embed(xref) $pos] 2]
}

# If we still can't figure out what to use, give up.

if {[info exists pluginPath]} {
    error "No plugin defined for $args"
}

# Load the required plugin.
# Save it's handle in a state variable keyed by the Mime Type

set pluginID $Embed(TYPE.$id)

if {[info exists Embed($pluginID)]} {
    set Embed($pluginID) [plugext create TST2 -path $pluginPath]
}

foreach a $args {
    if {[string first "=" $a] > 0} {
        if {[info exists embedargs]} {
            append embedargs " " $a
        } else {
            set embedargs $a
        }
    }
}
}

```



```

set Embed(instance.$id) [$Embed($pluginID) new \
    -mime $Embed(TYPE.$id) -embed $embedargs \
    -mode $mode]

#
# Generate a window, if HEIGHT and WIDTH are set, and HIDDEN=false/0
# If HIDDEN == True but HEIGHT/WIDTH are missing, set default size
# 200x200

# Can't use ! with a string, so I kludge to get !hidden
if {"$Embed(HIDDEN.$id)"} {} else {
    set ht 200;
    set wd 200;

    if {[info exists Embed(HEIGHT.$id)]} {
        set ht $Embed(HEIGHT.$id)
    }

    if {[info exists Embed(WIDTH.$id)]} {
        set wd $Embed(WIDTH.$id)
    }

    set Embed(canvas.$id) [canvas $Embed(CanvasParent).cPLUGIN$id \
        -height $ht -width $wd -background yellow]

    puts $output "CANVAS: $Embed(canvas.$id) :: $ht x $wd"; flush $output

    # Grid and update, to make sure the base OS has really allocated
    # space for this window before sending it to the plugin. Else,
    # life gets exciting.

    grid $Embed(canvas.$id) -row 0 -column 0
    update; update idle;

    $Embed(instance.$id) setwindow $Embed(canvas.$id)
    grid forget $Embed(canvas.$id)
}

#
# Figure out the type of stream we'll need to interact with
# Then close the stream again.
#

set Embed(out.$id) [$Embed(instance.$id) open -url file:$Embed(SRC.$id) -mime
$Embed(TYPE.$id) w]
fconfigure $Embed(out.$id) -translation binary
set streamType [$Embed(instance.$id) streamtype]

puts $output "STR: $streamType" ; flush $output

#
# Dump data as required by the stream type.
#

switch $streamType {
    "asfile" {

```

```

        $Embed(instance.$id) streamfile $Embed(SRC.$id)
    }
    "asfileonly" {
        $Embed(instance.$id) streamfile $Embed(SRC.$id)
    }
    "normal" {
        set in [open $Embed(SRC.$id) r]
        fconfigure $in -translation binary

        set maxTransfer [$Embed(instance.$id) writeready $Embed(out.$id)]
        if {$maxTransfer > 64000} {
            set maxTransfer 64000
        }

        puts $output "maxTransfer: $maxTransfer"
        fconfigure $Embed(out.$id) -buffersize $maxTransfer

        # If the file is small go for speed.
        # If it's greater than 64K, reduce the memory footprint,
        # since it won't be fast no matter what we do.

        if {[file size $Embed(SRC.$id)] > 64000} {
            while {[eof $in]} {
                set dat [read $in 4096]
                puts -nonewline $Embed(out.$id) $dat
            }
            flush $Embed(out.$id)
        } else {
            set dat [read $in]
            close $in
            puts -nonewline $Embed(out.$id) $dat
            flush $Embed(out.$id)
        }
        puts "DONE XFER"
    }
    default {
        error "Unrecognized stream value: $streamType"
    }
}
if {!$Embed(KEEPOPEN.$id)} {
    puts "CLOSING $Embed(out.$id)"
    closeFile $id
}
return $id
}

proc resetWindow {id} {
    variable Embed
    if {"$Embed(HIDDEN.$id)"} {} else {
        $Embed(instance.$id) setwindow $Embed(canvas.$id)
    }
}

proc closeFile {id} {
    variable Embed
    if {[info exists Embed(out.$id)]} {
        close $Embed(out.$id)
        unset Embed(out.$id)
    }
}

```